(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 March 2002 (21.03.2002)

PCT

(10) International Publication Number
WO 02/23362 A1

(51) International Patent Classification[7]: G06F 15/16

(21) International Application Number: PCT/US01/28391

(22) International Filing Date: 12 September 2001 (12.09.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/660,500 12 September 2000 (12.09.2000) US
60/274,615 12 March 2001 (12.03.2001) US

(71) Applicant (for all designated States except US): NETMOTION WIRELESS, INC. [US/US]; 1500 Dexter Avenue North, Seattle, WA 98109-3032 (US).

(72) Inventors; and
(75) Inventors/Applicants (for US only): HANSON, Aaron, D. [US/US]; 3002 NW 63rd Street, Seattle, WA 98107 (US). STURNIOLO, Emil, A. [US/US]; 4050 Alameda Court, Medina, OH 44256 (US). MENN, Anatoly [US/US]; 816 North 175th, #4, Seattle, WA 98133 (US). OLSON, Erik, D. [US/US]; 306 NWS 82nd Street, Seattle, WA 98117 (US). SAVARESE, Joseph, T. [US/US]; 22205 95th Place West, Edmonds, WA 98020 (US).

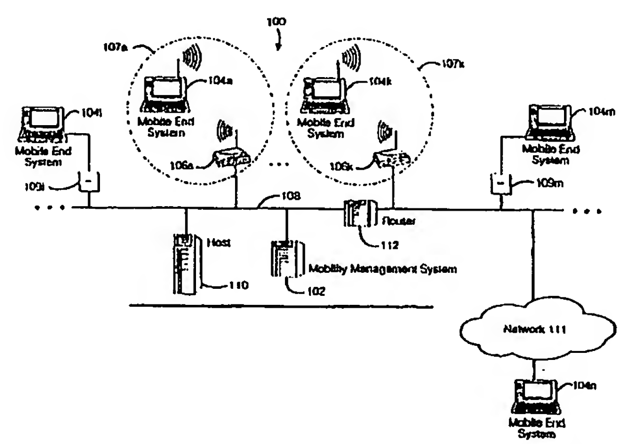
(74) Agent: FARIS, Robert, W.; Nixon & Vanderhye, 1100 North Glebe Road, Suite 800, Arlington, VA 22201-4714 (US).

(81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR PROVIDING MOBILE AND OTHER INTERMITTENT CONNECTIVITY IN A COMPUTING ENVIRONMENT



WO 02/23362 A1

(57) Abstract: A seamless solution transparently addresses the characteristics of nomadic systems, and enables existing network applications to run reliably in mobile environments. A Mobility Management Server (102) coupled to the mobile network maintains the state of each af any number of Mobile End Systems (104) and handles the complex session management required to maintain persistent connections to the network and to other peer processes. If a Mobile End System becomes unreachable, suspends, or changes network address (e.g., due to roaming from one network interconnect to another), the Mobility Management Server maintains the connection to the associated peer task- allowing the Mobile End System to maintain a continous connection even though it may temporarily lose contact with its network medium. An interface-based listener uses network point of attachment information supplied by a network interface to determine roaming conditions and to efficiently establish connection upon roaming. The Mobility Management Server can distribute lists to Mobile End Systems specifying how to contact it over disjoint networks.

Published:
- *with international search report*
- *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

METHOD AND APPARATUS FOR PROVIDING MOBILE AND OTHER INTERMITTENT CONNECTIVITY IN A COMPUTING ENVIRONMENT

FIELD OF THE INVENTION

The present invention relates to connectivity between networked computing devices. More particularly, the present invention relates to methods and systems that transparently address the characteristics of nomadic systems, and enable existing network applications to run reliably in the associated mobile environments. Still more particularly, the invention relates to techniques and systems for providing a continuous data stream connection between intermittently-connected devices such as handheld data units and personal computing devices.

BACKGROUND AND SUMMARY OF THE INVENTION

Increasingly, companies are seeing rapid access to key information as the way to maintaining a competitive advantage. To provide immediate access to this information, mobile and other intermittently-connected computing devices are quietly and swiftly becoming an essential part of corporate networks -- especially with the proliferation of inexpensive laptops and hand-held computing devices. However, integrating these nomadic devices into existing network infrastructures has created a challenge for the information manager.

Many problems in mobile networking parallel the difficulties in early local area networks (LANs) before the adoption of Ethernet. There are a variety of mobile protocols and interfaces, and because standards are just developing, there is little interoperability between systems. In addition, performance over these network technologies has typically been slow and

bandwidth limited. Implementation costs to date have been high due the specialized nature of deployed systems.

Along with these issues, mobile technologies present a category of problems unto their own. Interconnects back into the main network may travel over and through a public network infrastructure, thus allowing sensitive information to possibly be tapped into. Furthermore, if any of the intermediary interconnects are via a wireless interface, the information is actually broadcast, and anyone with a similar interface can eavesdrop without much difficulty.

But, perhaps even more significantly, mobile networking has generally in the past been limited to mostly message-oriented or stateless applications -- and thus has not been readily adaptable for existing or new corporate applications that use client/server, host-terminal, web-based or shared file systems models. This is because such commonly used applications need stateful sessions that employ a continuous stream of data -- not just a stateless packet exchange -- to work effectively and reliably.

To this end, many or most popular off-the-shelf networking applications require TCP/IP sessions, or private virtual circuits. These sessions cannot continue to function if they encounter network interruptions, nor can they tolerate roaming between networks (i.e., a change of network addresses) while established. Yet, mobile networking is, by its nature, dynamic and unreliable. Consider these common scenarios encountered in mobile networks:

Disconnected or Out of Range User

When a mobile device disconnects from a given network or loses contact (e.g., through an outage or "hole" in the coverage of a wireless interconnect), the session-oriented application running on the mobile device

loses its stateful connection with its peer and ceases to operate. When the device is reattached or moves back into contact, the user must re-connect, log in again for security purposes, find the place in the application where work was left off, and possibly re-enter lost data. This reconnection process is time consuming, costly, and can be very frustrating.

Moving to a Different Network or Across a Router Boundary (Network Address Change)

Mobile networks are generally segmented for manageability purposes. But the intent of mobile devices is to allow them to roam. Roaming from one network interconnect to another can mean a change of network address. If this happens while the system is operational, the routing information must be changed for communications to continue between the associated peers. Furthermore, acquiring a new network address may require all of the previously established stateful application sessions to be terminated -- again presenting the reconnection problems noted above.

Security

As mentioned before, companies need to protect critical corporate data. Off-the-shelf enterprise applications are often written with the assumption that access to the physical network is controlled (i.e., carried within cables installed inside a secure facility), and security is maintained through an additional layer of authentication and possible encryption. These assumptions have not been true in the nomadic computing world -- where data is at risk for interception as it travels over public airways or public wire-line infrastructures.

It would be highly desirable to provide an integrated solution that transparently addresses the characteristics of nomadic systems, and enables existing network applications to run reliably in these mobile environments.

The present invention solves this problem by providing a seamless solution that extends the enterprise network, letting network managers provide mobile users with easy access to the same applications as stationary users without sacrificing reliability or centralized management. The solution combines advantages of present-day wire-line network standards with emerging mobile standards to create a solution that works with existing network applications.

In accordance with one aspect of a non-limiting exemplary and illustrative embodiment of the present invention, a Mobility Management Server (MMS) coupled to the mobile interconnect maintains the state of each of any number of Mobile End Systems (MES) and handles the complex session management required to maintain persistent connections to the network and to peer application processes. If a Mobile End System becomes unreachable, suspends, or changes network address (e.g., due to roaming from one network interconnect to another), the Mobility Management Server maintains the connection to the associated peer -- allowing the Mobile End System to maintain a continuous virtual connection even though it may temporarily lose its actual physical connection.

The illustrative non-limiting example embodiments provided by the present invention also provide the following (among others) new and advantageous techniques and arrangements:

- a Mobility Management Server providing user configurable session priorities for mobile clients;

- per-user mobile policy management for managing consumption of network resources;
- a roaming methodology making use of the industry standard Dynamic Host Configuration Protocol (DHCP) in coordination with a Mobility Management Server;
- automatic system removal of unreliable datagrams based on user configurable timeouts; and
- automatic system removal of unreliable datagrams based on user configurable retries.

In more detail, the preferred illustrative embodiments of the present invention in one of their aspects provide a Mobility Management Server that is coupled to the mobile interconnect (network). The Mobility Management Server maintains the state of each of any number of Mobile End Systems and handles the complex session management required to maintain persistent connections to the network and to other processes (e.g., running on other network-based peer systems). If a Mobile End System becomes unreachable, suspends, or changes network address (e.g., due to roaming from one network interconnect to another), the Mobility Management Server maintains the connection to the associated peer, by acknowledging receipt of data and queuing requests. This proxying by the Mobility Management Server allows the application on the Mobile End System to maintain a continuous connection even though it may temporarily lose its physical connection to a specific network medium.

In accordance with another aspect of preferred embodiments of the present invention, a Mobility Management Server manages addresses for Mobile End Systems. Each Mobile End System is provided with a proxy address on the primary network. This highly available address is known as the "virtual address" of the Mobile End System. The Mobility Management

Server maps the virtual addresses to current "point of presence" addresses of the nomadic systems. While the point of presence address of a Mobile End System may change when the mobile system changes from one network interconnect to another, the virtual address stays constant while any connections are active or longer if the address is statically assigned.

In accordance with yet another aspect of the preferred exemplary embodiments of the present invention, a Mobility Management Server provides centralized system management of Mobile End Systems through a console application and exhaustive metrics. The preferred embodiment also provides user configurable session priorities for mobile clients running through a proxy server, and per-user mobile policy management for managing consumption of network resources.

In accordance with yet another aspect, a Remote Procedure Call protocol and an Internet Mobility Protocol are used to establish communications between the proxy server and each Mobile End System.

Remote procedure calls provide a method for allowing a process on a local system to invoke a procedure on a remote system. The use of the RPC protocol allows Mobile End Systems to disconnect, go out of range or suspend operation without losing active network sessions. Since session maintenance does not depend on a customized application, off-the-shelf applications will run without modification in the nomadic environment.

The Remote Procedure Call protocol generates transactions into messages that can be sent via the standard network transport protocol and infrastructure. These RPC messages contain the entire network transaction initiated by an application running on the Mobile End System -- enabling the Mobility Management Server and Mobile End System to keep connection state information synchronized at all times -- even during interruptions of the physical link connecting the two. In the preferred

embodiment of the present invention providing RPC's, the proxy server and the Mobile End Systems share sufficient knowledge of each transaction's state to maintain coherent logical database about all shared connections at all times.

The Internet Mobility Protocol provided in accordance with the preferred embodiment of the present invention compensates for differences between wired local area network interconnects and other less reliable networks such as a wireless LAN or WAN. Adjusted frame sizes and protocol timing provide significant performance improvements over non-mobile-aware transports -- dramatically reducing network traffic. This is important when bandwidth is limited or when battery life is a concern. The Internet Mobility Protocol provided in accordance with the preferred embodiment of the present invention also ensures the security of organizational data as it passes between the Mobile End System and the Mobility Management Server over public network interconnects or airways. The Internet Mobility Protocol provides a basic firewall function by allowing only authenticated devices access to the organizational network. The Internet Mobility Protocol can also certify and encrypt all communications between the Mobility Management Server and the Mobile End System.

In accordance with yet another aspect of preferred non-limiting embodiments of the present invention, mobile inter-connectivity is built on standard transport protocols (e.g., TCP/IP, UDP/IP and DHCP, etc) to extend the reach of standard network application interfaces. The preferred exemplary embodiments of the present invention efficiently integrates transport, security, address management, device management and user management needs to make nomadic computing environments effectively transparent. The Internet Mobility Protocol provides an efficient

mechanism for multiplexing multiple streams of data (reliable and unreliable) through a single virtual channel provided by such standard transport protocols over standard network infrastructure.

With the help of the RPC layer, the Internet Mobility Protocol coalesces data from different sources targeted for the same or different destinations, together into a single stream and forwards it over a mobile link. At the other end of the mobile link, the data is demultiplexed back into multiple distinct streams, which are sent on to their ultimate destination(s). The multiplexing/demultiplexing technique allows for maximum use of available bandwidth (by generating the maximum sized network frames possible), and allows multiple channels to be established (thus allowing prioritization and possibly providing a guaranteed quality of service if the underlying network provides the service).

The Internet Mobility Protocol provided in accordance with the preferred example embodiments of the present invention provide the additional non-limiting exemplary features and advantages:

- Transport protocol independence.
- Allows the network point of presence (POP) or network infrastructure to change without affecting the flow of data (except where physical boundary, policy or limitations of bandwidth may apply).
- Minimal additional overhead.
- Automatic fragment resizing to accommodate the transmission medium. (When the protocol data unit for a given frame is greater then the available maximum transmission unit of the network medium, the Internet Mobility Protocol will fragment and reassemble the frame to insure that it can traverse the network. In the event of a retransmit, the frame will again be

assessed. If the network infrastructure or environment changes, the frame will be refragmented or in the case that the maximum transmission unit actually grew, sent as a single frame.)

- Semantics of unreliable data are preserved, by allowing frames to discard unreliable data during retransmit.
- Provides a new semantic of Reliable Datagram service. (Delivery of datagrams can now be guaranteed to the peer terminus of the Internet Mobility Protocol connection. Notification of delivery can be provided to a requesting entity.)
- Considers the send and receive transmission path separately, and automatically tailors its operating parameters to provided optimum throughput. (Based on hysteresis, it adjusts such parameters as frame size/fragmentation threshold, number of frames outstanding (window), retransmit time, and delayed acknowledgement time to reduce the amount of duplicate data sent through the network.)
- Network fault tolerant (since the expected usage is in a mobile environment, temporary loss of network medium connectivity does not result in a termination of the virtual channel or application based connection).
- Provides an in-band signaling method to its peer to adjust operating parameters (each end of the connection can alert its peer to any changes in network topology or environment).
- Employs congestion avoidance algorithms and gracefully decays throughput when necessary.
- Employs selective acknowledgement and fast retransmit policies to limit the number of gratuitous retransmissions, and provide faster handoff recovery in nomadic environments. (This also

allows the protocol to maintain optimum throughput in a lossy network environment.)

- Employs sliding window technology to allow multiple frames to be outstanding. (This parameter is adjustable in each direction and provides for streaming frames up to a specified limit without requiring an acknowledgement from its peer.)
- Sequence numbers are not byte oriented, thus allowing for a single sequence number to represent up to a maximum payload size.
- Security aware. (Allows for authentication layer and encryption layer to be added in at the Internet Mobility Protocol layer.)
- Compression to allow for better efficiency through bandwidth limited links.
- Balanced design, allowing either peer to migrate to a new point of presence.
- Either side may establish a connection to the peer.
- Allows for inactivity timeouts to be invoked to readily discard dormant connections and recover expended resources.
- Allows for a maximum lifetime of a given connection (e.g., to allow termination and/or refusal to accept connections after a given period or time of day).

Non-limiting preferred exemplary embodiments of the present invention also allow a system administrator to manage consumption of network resources. For example, the system administrator can place controls on Mobile End Systems, the Mobility Management Server, or both. Such controls can be for the purpose, for example, of managing allocation of network bandwidth or other resources, or they may be related to security

issues. It may be most efficient to perform management tasks at the client side for clients with lots of resources. However, thin clients don't have many resources to spare, so it may not be practical to burden them with additional code and processes for performing policy management. Accordingly, it may be most practical to perform or share such policy management functions for thin clients at a centralized point such as the Mobility Management Server. Since the Mobility Management Server proxies the distinct data streams of the Mobile End Systems, it provides a central point from which to conduct policy management. Moreover, the Mobility Management Server provides the opportunity to perform policy management of Mobile End Systems on a per user and/or per device basis. Since the Mobility Management Server is proxying on a per user basis, it has the ability to control and limit each user's access to network resources on a per-user basis as well as on a per-device basis.

As one simple example, the Mobility Management Server can "lock out" certain users from accessing certain network resources. This is especially important considering that interface network is via a mobile interconnect, and may thus "extend" outside of the boundaries of a locked organizational facility (consider, for example, an ex-employee who tries to access the network from outside his former employer's building). However, the policy management provided by the Mobility Management Server can be much more sophisticated. For example, it is possible for the Mobility Management Server to control particular Web URL's particular users can visit, filter data returned by network services requests, and/or compress data for network bandwidth conservation. This provides a way to enhance existing and new application-level services in a seamless and transparent manner.

Furthermore, because the Mobile End System may be connected to an "untrusted" network (i.e. outside the corporations locked boundaries) there is a chance of malicious attack while being remotely connected. By sharing policy rules and filters with the Mobile End System, one can protect the MES from rogue connections, provide ingress filtering for the remote node, and further secure the corporate infrastructure from one central location.

A further exemplary embodiment of the invention provides an interface-assisted roaming listener that allows Mobile End Systems to take advantage of interfaces supporting generic signaling, to enable interface-assisted roaming. In accordance with one feature provided in accordance with the exemplary embodiment, the Mobile End System interface-based listener determines in response to an event (e.g., a callback, a timer timeout or a network activity hint indicating data loss), whether the Mobile End System's media connectivity status has changed. For example, the listener signals to the interface when it detects that the Mobile End System has become detached and is no longer in communication with the network. Upon re-attachment, the listener uses previously recorded network point of attachment identification information to determine whether the Mobile End System has been reattached to the same or different network point of attachment. If the reattachment is to the same network point of attachment, the listener signals to alert the mobile clients that they need to take steps to reestablish transport level communications. If the reattachment is to a different network point of attachment, the listener signals a "roam" condition and prompts the Mobile End System to acquire an address that is usable on the current network segment (this may entail, for example, registering the current address to be valid on a new subnet, for example). The listener may maintain a network topology map (which may be learned

through operation) to assist it in deciding the correct signal (e.g., "roam same subnet" or "roam") to generate to its clients.

A still further aspect provided by non-limiting preferred exemplary embodiments of our invention allows access to a Mobility Management Server (MMS) in a "disjoint networking" mode. The new algorithm allows for dynamic/static discovery of alternate network addresses that can be used to establish/continue communications with an MMS -- even in a disjoint network topology in which one network infrastructure may have no knowledge of network addresses for another network infrastructure. In accordance with this arrangement, a list of alternate addresses that the MMS is available at is preconfigured, forwarded to or dynamically learned by an MES (Mobile End System) during the course of a conversation/connection. In one embodiment, the MMS can use a connection over one network to send the MES one or more MMS network addresses or other MMS identities corresponding to other networks. This list can be sent/updated during circuit creation or at any other time during the connection.

If/when the MES roams to a second network, it uses the list of MMS "alias" addresses/identifications to contact the MMS over the new network connection on the second network. This allows the MES to re-establish contact with the MMS over the new network connection even though the first and second networks may not share any addresses, routes, or other information.

Further example embodiments of the invention provide policy management decision making within a distributed mobile network topology. For example, rule-based policy management procedures can be performed to allow, deny and/or condition request fulfillment based on a variety of metrics. Such policy management can be used, for example, to provide

decision making based on cost metrics such as least cost routing in a multi-home/path environment.

Such policy management techniques may take into account the issue of mobility or positioning (i.e., roaming) in making decisions. For example, policy management rules may be based on locale of the device (e.g., proximity to network point of attachment such as access point/base station, hubs, routers, or GPS coordinate) so certain operations can be allowed in one building of an enterprise but not in another building. An example of such an application might be an enterprise with several different wireless networks. Such an enterprise might have a loading dock and office area served by a wireless network. The system administrator would be able to configure the system such that workers (e.g., users and devices) on the loading dock are not permitted access to the wireless network in the office environment. Also policy management results can be tri-state: allow, deny or delay (for example, if the decision is based on bandwidth requirements or cost, the decision may be to delay an operation from being executed and to wait for a more opportune time when the operation can be accommodated).

The policy management provided by the preferred example embodiments is capable of modifying an operation based on a decision. For example, one example action is to throttle consumption of network bandwidth for all active applications. Also consider an aggressive application that is consuming significant bandwidth. The policy engine can govern the rate at which the application's operations/transactions are completed. This behavior may also be learned dynamically to squelch a possible errant application. Another example action provides reconstitution of data(i.e. dithering of graphics images based on available/allowable bandwidth or cost/user restrictions).

Furthermore the rules engine allows for other actions to be invoked based on rule evaluation. External procedures such as logging an event, sending an alert or notifying the user that the action is being denied, delayed, or conditioned may be executed. Such notification can also be interactive and ask for possible overrides to an existing rule from the operator.

The policy management engine provided in the example non-limiting embodiment can base its decision making on any number or combination metrics that are associated with the device, device group, user group, user,/or network point of attachment.

As part of the policy management functionality, other locale base information and services may also be acquired/provided for the purposes of policy management, network modeling, and/or asset tracking. Such services include the ability to automatically present to users and mobile end systems information that is applicable within the context of a mobile end system's present location. This information may be provided in the form of messages, files, or in some other electronic format.

One non-limiting example of such use of this capability would permit shopping malls, business communities, and large retailers, to locate wireless access points that may be compatible with Bluetooth PANs, IEEE 802.11 LANs, 802.15 PANs, or other wireless network standards in strategic locations within the shopping center. As customers roam from location to location, stores and vendors would be permitted to push down information relevant to the vendors that are present within the mobile end systems current location. This information would include information such as current sales, discounts, and services. In addition to such information, mobile end systems may be provided electronic coupons used for sales promotion. Vendors would be permitted to register for these services

through some centralized authority that may be associated with the mall, business community, retailer, or some other hosted service.

A further example non-limiting use of such a technology would be in vertical industries where information is collected based on location including but not limited to such industries as field service, field sales, package delivery, or public safety where lists of local services, maps, directions, customers, or hazards may be pushed down to mobile end systems based on location.

Yet another non-limiting example use may entail a web based service for monitoring and tracking mobile end systems. For example, customers may register for this tracking service so trusted third parties may log onto the hosted web service and find out exact locations of their mobile end systems. This may include mobile end systems installed on vehicles or carried by pedestrians. As mobile end systems continue to experience reductions in size and wait, such services become more likely. These services would permit people to track and locate individuals that are high risk such as elderly, handicapped, or ill. It may also be used to track items that are highly valued such as automobiles or other expensive mobile properties and packages. Using 3G WAN networks, Bluetooth networks, 802.11 networks, 802.15 networks, and other wireless technologies, combined with this unique ability to provide seamless connectivity while switching network mediums or point of attachments, such services become possible and likely at a much reduced cost.

The present invention thus extends the enterprise network, letting network managers provide mobile users with easy access to the same applications as stationary users without sacrificing reliability or centralized management. The solution combines advantages of existing wire-line

network standards with emerging mobility standards to create a solution that works with existing network applications.

BRIEF DESCRIPTION OF THE DRAWINGS

These, as well as other features and advantages of this invention, will be more completely understood and appreciated by careful study of the following more detailed description of presently preferred example embodiments of the invention taken in conjunction with the accompanying drawings, of which:

Figure 1 is a diagram of an overall mobile computing network provided in accordance with the present invention;

Figure 2 shows an example software architecture for a Mobile End System and a Mobility Management Server;

Figure 2A shows example steps performed to transfer information between a Mobile End System and a Mobility Management Server;

Figure 3 shows an example mobile interceptor architecture;

Figure 3A is a flowchart of example steps performed by the mobile interceptor;

Figure 3B is a flowchart of example steps performed by an RPC engine to handle RPC work requests;

Figures 4-5C are flowcharts of example steps to process RPC work requests;

Figure 6 is a diagram of an example received work request;

Figure 7 is a diagram showing how a received work request can be dispatched onto different priority queues;

Figures 8 and 9 show processing of the contents of the different priority queues;

Figures 10A-15B show example steps performed to provide an Internet Mobility Protocol;

Figure 16 shows example listener data structures;

Figures 17, 17A and 18 are flowcharts of example steps performed to provide for mobile interconnect roaming;

Figures 19A and 19B are together a flowchart of an example interface-assisted roaming process;

Figure 20 shows an example interface assisted roaming topology node data structure;

Figure 21 shows an example technique for distributing mobility management system network addresses to mobile end systems in a disjoint network topology;

Figure 22 shows an example use of the Figure 21 technique to provide secure communications;

Figure 23 shows an example use of the Figure 21 technique to provide network address translation in a distributed network interface scenario;

Figure 24 shows an example policy management table; and

Figure 25 is a flowchart of example policy management processing steps

DETAILED DESCRIPTION OF NON-LIMITING PRESENTLY PREFERRED EXAMPLE EMBODIMENTS

Figure 1 is an example of mobile enhanced networked computer system 100 provided in accordance with the present invention. Networked computer system 100 includes a Mobility Management Server 102 and one or more Mobile End Systems 104. Mobile End Systems 104 can communicate with Mobility Management Server 102 via a local area

network (LAN) 108. Mobility Management Server 102 serves as network level proxy for Mobile End Systems 104 by maintaining the state of each Mobile End System, and by handling the complex session management required to maintain persistent connections to any peer systems 110 that host network applications -- despite the interconnect between Mobile End Systems 104 and Mobility Management Server 102 being intermittent and unreliable. In the preferred embodiment, Mobility Management Server 102 communicates with Mobile End Systems 104 using Remote Procedure Call and Internet Mobility Protocols in accordance with the present invention.

In this particular example, Mobile End Systems 104 are sometimes but not always actively connected to Mobility Management Server 102. For example:

- Some Mobile End Systems 104a-104k may communicate with Mobility Management Server 102 via a mobile interconnect (wirelessly in this case), e.g., conventional electromagnetic (e.g., radio frequency) transceivers 106 coupled to wireless (or wire-line) local area or wide area network 108. Such mobile interconnect may allow Mobile End Systems 104a-104k to "roam" from one cover area 107a to another coverage area 107k. Typically, there is a temporary loss of communications when a Mobile End System 104 roams from one coverage area 107 to another, moves out of range of the closest transceiver 106, or has its signal temporarily obstructed (e.g., when temporarily moved behind a building column or the like).
- Other Mobile End Systems 104l, 104m, ... may communicate with Mobility Management Server 102 via non-permanent wire-based interconnects 109 such as docking ports, network cable connectors, or the like. There may be a temporary loss of communications when Mobile End Systems 104 are temporarily

disconnected from LAN 108 by breaking connection 109, powering off the Mobile End Systems, etc.

- Still other Mobile End Systems (e.g., 104n) may be nomadically coupled to Mobility Management Server 102 via a further network topography 111 such as a wide area network, a dial-up network, a satellite network, or the Internet, to name a few examples. In one example, network 111 may provide intermittent service. In another example, Mobile End Systems 104 may move from one type of connection to another (e.g., from being connected to Mobility Management Server 102 via wire-based interconnect 109 to being connected via network 111, or vice versa) -- its connection being temporarily broken during the time it is being moved from one connection to another.

Mobile End Systems 104 may be standard mobile devices and off the shelf computers. For example, Mobile End System 104 may comprise a laptop computer equipped with a conventional radio transceiver and/or network cards available from a number of manufacturers. Mobile End Systems 104 may run standard network applications and a standard operating system, and communicate on the transport layer using a conventionally available suite of transport level protocols (e.g., TCP/IP suite.) In accordance with the present invention, Mobile End Systems 104 also execute client software that enables them to communicate with Mobility Management Server 102 using Remote Procedure Call and Internet Mobility Protocols that are transported using the same such standard transport level protocols.

Mobility Management Server 102 may comprise software hosted by a conventional Windows NT or other server. In the preferred embodiment, Mobility Management Server 102 is a standards-compliant, client-server

based intelligent server that transparently extends the enterprise network 108 to a nomadic environment. Mobility Management Server 102 serves as network level proxy for each of any number of Mobile End Systems 104 by maintaining the state of each Mobile End System, and by handling the complex session management required to maintain persistent connections to any peer systems 110 that host network applications -- despite the mobile interconnect between Mobile End Systems 104 and transceivers 106 being intermittent and unreliable.

For example, server 102 allows any conventional (e.g., TCP/IP based) network application to operate without modification over mobile connection. Server 102 maintains the sessions of Mobile End Systems 104 that disconnect, go out of range or suspend operation, and resumes the sessions when the Mobile End System returns to service. When a Mobile End System 104 becomes unreachable, shuts down or changes its point of presence address, the Mobility Management Server 102 maintains the connection to the peer system 110 by acknowledging receipt of data and queuing requests until the Mobile End System once again becomes available and reachable.

Server 102 also extends the management capabilities of wired networks to mobile connections. Each network software layer operates independently of others, so the solution can be customized to the environment where it is deployed.

As one example, Mobility Management Server 102 may be attached to a conventional organizational network 108 such as a local area network or wide area network. Network 108 may be connected to a variety of fixed-end systems 110 (e.g., one or most host computers 110). Mobility Management Server 102 enables Mobile End Systems 104 to communicate with Fixed End System(s) 110 using continuous session type data streams

even though Mobile End Systems 104 sometimes lose contact with their associated network interconnect or move from one network interconnect 106, 109, 111 to another (e.g., in the case of wireless interconnect, by roaming from one wireless transceiver 106 coverage area 107 to another).

A Mobile End System 104 establishes an association with the Mobility Management Server 102, either at startup or when the Mobile End System requires network services. Once this association is established, the Mobile End System 104 can start one or more network application sessions, either serially or concurrently. The Mobile End System 104-to-Mobility Management Server 102 association allows the Mobile End System to maintain application sessions when the Mobile End System, disconnects, goes out of range or suspends operation, and resume sessions when the Mobile End System returns to service. In the preferred embodiment, this process is entirely automatic and does not require any intervention on the user's part.

In accordance with an aspect of the present invention, Mobile End Systems 104 communicate with Mobility Management Server 102 using conventional transport protocols such as, for example, UDP/IP. Use of conventional transport protocols allows Mobile End Systems 104 to communicate with Mobility Management Server 102 using the conventional routers 112 and other infrastructure already existing on organization's network 108. In accordance with the present invention, a higher-level Remote Procedure Call protocol generates transactions into messages that are sent over the mobile enhanced network 108 via the standard transport protocol(s). In this preferred embodiment, these mobile RPC messages contain the entire network transaction initiated by an application running on the Mobile End System 104, so it can be completed in its entirety by the Mobility Management Server. This enables the Mobility Management

Server 102 and Mobile End System 104 to keep connection state information synchronized at all times -- even during interruptions of network medium connectivity.

Each of Mobile End Systems 104 executes a mobility management software client that supplies the Mobile End System with the intelligence to intercept all network activity and relay it via the mobile RPC protocol to Mobility Management Server 102. In the preferred embodiment, the mobility management client works transparently with operating system features present on Mobile End Systems 104 (e.g., Windows NT, Windows 98, Windows 95, Windows CE, etc.) to keep client-site application sessions active when contact is lost with the network.

Mobility Management Server 102 maintains the state of each Mobile End System 104 and handles the complex session management required to maintain persistent connections to associated peer 108 such as host computer 110 attached to the other end of the connection end point. If a Mobile End System 104 becomes unreachable, suspends, or changes network address (e.g., due to roaming from one network interconnect to another), the Mobility Management Server 102 maintains the connection to the host system 110 or other connection end-point, by acknowledging receipt of data and queuing requests. This proxy function means that the peer application never detects that the physical connection to the Mobile End System 104 has been lost -- allowing the Mobile End System's application(s) to effectively maintain a continuous connection with its associated session end point (by simply and easily resuming operations once a physical connection again is established) despite the mobile system temporarily losing connection or roaming from one network interconnect 106A to another network interconnect 106K within coverage area 107K.

Mobility Management Server 102 also provides address management to solve the problem of Mobile End Systems 104 receiving different network addresses when they roam to different parts of the segmented network. Each Mobile End System 104 is provided with a virtual address on the primary network. Standard protocols or static assignment determine these virtual addresses. For each active Mobile End System 104, Mobility Management Server 102 maps the virtual address to the Mobile End System's current actual ("point of presence") address. While the point of presence address of a Mobile End System 104 may change when the device changes from one network segment to another, the virtual address stays constant while any connections are active or longer if the address is assigned statically.

Thus, the change of a point of presence address of a Mobile End System 104 remains entirely transparent to an associated session end point on host system 110 (or other peer) communicating with the Mobile End System via the Mobility Management Server 102. The peer 110 sees only the (unchanging) virtual address proxied by the server 102.

In the preferred embodiment, Mobility Management Server 102 can also provide centralized system management through console applications and exhaustive metrics. A system administrator can use these tools to configure and manage remote connections, and troubleshoot remote connection and system problems.

The proxy server function provided by Mobility Management Server 102 allows for different priority levels for network applications, users and machines. This is useful because each Mobility Management Server 102 is composed of finite processing resources. Allowing the system manager to configure the Mobility Management Server 102 in this way provides enhanced overall system and network performance. As one example, the

system manager can configure Mobility Management Server 102 to allow real time applications such as streaming audio or video to have greater access to the Mobility Management Server 102's resources than other less demanding applications such as email.

In more detail, Mobility Management Server 102 can be configured via an application or application interface; standard network management protocols such as SNMP; a Web-based configuration interface; or a local user interface. It is possible to configure association priority and/or to configure application priority within an association. For example, the priority of each association relative to other associations running through the Mobility Management Server 102 is configurable by either the user name, or machine name (in the preferred embodiment, when the priority is configured for both the user and the machine that a user is logged in on, the configuration for the user may have higher precedence). In addition or alternatively, each association may have several levels of application priority, which is configured based on network application name. The system allows for any number of priority levels to exist. In one particular implementation, three priority levels are provided: low, medium and high.

Server and Client Example Software Architecture

Figure 2 shows an example software architecture for Mobile End System 104 and Mobility Management Server 102. In accordance with one aspect of the present invention, Mobile End System 104 and Mobility Management Server 102 run standard operating system and application software -- with only a few new components being added to enable reliable and efficient persistent session connections over an intermittently connected mobile network 108. As shown in Figure 2, Mobile End System 104 runs conventional operating system software including network interface drivers

200, TCP/UDP transport support 202, a transport driver interface (TDI) 204, and a socket API 206 used to interface with one or more conventional network applications 208. Conventional network file and print services 210 may also be provided to communicate with conventional TDI 204. Server 102 may include similar conventional network interface drivers 200', TCP/UDP transport support 202', a transport driver interface (TDI) 204', and a socket API 206' used to interface with one or more conventional network applications 208'. Mobile End System 104 and Mobility Management Server 102 may each further include conventional security software such as a network/security provider 236 (Mobile End System) and a user/security database 238 (server).

In accordance with the present invention, a new, mobile interceptor component 212 is inserted between the TCP/UDP transport module 202 and the transport driver interface (TDI) 204 of the Mobile End System 104 software architecture. Mobile interceptor 212 intercepts certain calls at the TDI 204 interface and routes them via RPC and Internet Mobility Protocols and the standard TCP/UDP transport protocols 202 to Mobility Management Server 102 over network 108. Mobile interceptor 212 thus can intercept all network activity and relay it to server 102. Interceptor 212 works transparently with operating system features to allow client-side application sessions to remain active when the Mobile End System 104 loses contact with network 108.

While mobile interceptor 212 could operate at a different level than the transport driver interface 204 (e.g., at the socket API level 206), there are advantages in having mobile interceptor 212 operate at the TDI level or more specifically, any transport protocol interface. For brevity sake, all references to the transport driver interface will be denoted using the acronym TDI. Many conventional operating systems (e.g., Microsoft

Windows 95, Windows 98, Windows NT and Windows CE, etc.) provide TDI interface 204 -- thus providing compatibility without any need to change operating system components. Furthermore, because the transport driver interface 204 is normally a kernel level interface, there is no need to switch to user mode -- thus realizing performance improvements. Furthermore, mobile interceptor 212 working at the level of TDI interface 204 is able to intercept from a variety of different network applications 208 (e.g., multiple simultaneously running applications) as well as encompassing network file, print and other kernel mode services 210 (which would have to be handled differently if the interceptor operated at the socket API level 206 for example).

Figure 2A shows an example high level flowchart of how mobile interceptor 212 works. A call to the TDI interface 204 of Mobile End System 104 (block 250) is intercepted by mobile interceptor 212 (block 252). Mobile interceptor 212 packages the intercepted RPC call in a fragment in accordance with an Internet Mobility Protocol, and sends the fragment as a datagram via a conventional transport protocol such as UDP or TCP over the LAN, WAN or other transport 108 to Mobility Management Server 102 (block 252). The Mobility Management Server 102 receives and unpackages the RPC datagram (block 254), and provides the requested service (for example, acting as a proxy to the Mobile End System application 208 by passing data or a response to a application server process running on Fixed End System 110).

Referring once again to Figure 2, Mobility Management Server 102 includes an address translator 220 that intercepts messages to/from Mobile End Systems 104 via a conventional network interface driver 222. For example, address translator 230 recognizes messages from an associated session peer (Fixed End System 110) destined for the Mobile End System

104 virtual address. These incoming Mobile End System messages are provided to proxy server 224, which then maps the virtual address and message to previously queued transactions and then forwards the responses back to the current point of presence addresses being used by the associated Mobile End System 104.

As also shown in Figure 2, Mobility Management Server 102 includes, in addition to address translation (intermediate driver) 220, and proxy server 224, a configuration manager 228, a control/user interface 230 and a monitor 232. Configuration management 228 is used to provide configuration information and parameters to allow proxy server 224 to manage connections. Control, user interface 230 and monitor 232 allow a user to interact with proxy server 224.

<u>Mobile Interceptor</u>

Figure 3 shows an example software architecture for mobile interceptor 212 that support the RPC Protocol and the Internet Mobility Protocol in accordance with the present invention. In this example, mobile interceptor 212 has two functional components:

a Remote Procedure Call protocol engine 240; and

an Internet Mobility Protocol engine 244.

Proxy server 224 running on Mobility Management Server 102 provides corresponding engines 240', 244'.

Mobile interceptor 212 in the preferred embodiment thus supports Remote Procedure Call protocol and Internet Mobility Protocol to connect Mobility Management Server 102 to each Mobile End Systems 104. Remote procedure calls provide a method for allowing a process on a local system to invoke a procedure on a remote system. Typically, the local system is not aware that the procedure call is being executed on a remote

system. The use of RPC protocols allows Mobile End Systems 104 to go out of range or suspend operation without losing active network sessions. Since session maintenance does not depend on a customized application, off-the-shelf applications will run without modification in the mobile environment of network 108.

Network applications typically use application-level interfaces such as Windows sockets. A single call to an application-level API may generate several outgoing or incoming data packets at the transport, or media access layer. In prior mobile networks, if one of these packets is lost, the state of the entire connection may become ambiguous and the session must be dropped. In the preferred embodiment of the present invention providing RPCs, the Mobility Management Server 102 and the Mobile End Systems 104 share sufficient knowledge of the connection state to maintain a coherent logical link at all times -- even during physical interruption.

The Internet Mobility Protocol provided in accordance with the present invention compensates for differences between wire-line and other less reliable networks such as wireless. Adjusted frame sizes and protocol timing provide significant performance improvements over non-mobile-aware transports -- dramatically reducing network traffic. This is important when bandwidth is limited or when battery life is a concern.

The Internet Mobility Protocol provided in accordance with the present invention also ensure the security of organization's data as it passes between the Mobile End System 104 and the Mobility Management Server 102 on the public wire-line networks or airway. The Internet Mobility Protocol provides a basic firewall function by allowing only authenticated devices access to the organizational network. The Internet Mobility Protocol can also certify and encrypt all communications between the mobility management system 102 and the Mobile End System 104.

The Remote Procedure Call protocol engine 240 on Mobile End System 104 of Figure 3 marshals TDI call parameters, formats the data, and sends the request to the Internet Mobility Protocol engine 244 for forwarding to Mobility Management Server 102 where the TDI Remote Procedure Call engine 240' executes the calls. Mobile End Systems 104 martial TDI call parameters according to the Remote Procedure Call protocol. When the Mobility Management Server 102 TDI Remote Procedure Call protocol engine 240' receives these RPCs, it executes the calls on behalf of the Mobile End System 104. The Mobility Management Server 102 TDI Remote Procedure Call protocol engine 240' shares the complete network state for each connected Mobile End System with the peer Mobile End System 104's RPC engine 240. In addition to performing remote procedure calls on behalf of the Mobile End Systems 104, the server RPC engine 240' is also responsible for system flow control, remote procedure call parsing, virtual address multiplexing (in coordination with services provided by address translator 220), remote procedure call transaction prioritization, scheduling, policy enforcement, and coalescing.

The Internet Mobility Protocol engine 244 performs reliable datagram services, sequencing, fragmentation, and re-assembly of messages. It can, when configured, also provide authentication, certification, data encryption and compression for enhanced privacy, security and throughput. Because the Internet Mobility Protocol engine 244 functions in power-sensitive environments using several different transports, it is power management aware and is transport independent.

Figure 3A shows an example process mobile interceptor 212 performs to communicate a TDI call to Mobility Management Server 102. Generally, the mobile interceptor RPC protocol engine 240 forwards marshaled TDI calls to the Internet Mobility Protocol engine 244 to be

transmitted to the Mobility Management Server 102. RPC protocol engine 240 does this by posting the RPC call to a queue maintained by the Internet Mobility Protocol engine 244 (block 302). To facilitate bandwidth management, the Internet Mobility Protocol engine 244 delays sending received RPC calls for some period of time ("the RPC coalesce time out period") (block 304). Typically, the RPC coalesce timeout is set between five and fifteen milliseconds as one example but is user configurable. This delay allows the RPC engine 240 to continue posting TDI calls to the Internet Mobility Protocol engine 244 queue so that more than one RPC call can be transmitted to the Mobility Management Server 102 in the same datagram (fragment).

When the coalesce timer expires, or the RPC protocol engine 240 determines that it will not be receiving more RPC calls (decision block 306), the RPC engine provides the Internet Mobility Protocol engine 244 with a request to flush the queue, coalesce the RPC calls into a single frame, and forward the frame to its peer (block 308). This coalescing reduces the number of transmissions -- enhancing protocol performance. However, the Internet Mobility Protocol may also decide to flush queue 244 based on other external criteria to further optimize performance. In the preferred embodiment, if a single RPC request will fill an entire frame, the IMP layer will automatically try to send the request to the peer.

As mentioned above, Mobility Management Server 102 proxy server also has an RPC protocol engine 212' and an Internet Mobility Protocol engine 244'. Figure 3B shows an example process performed by Mobility Management Server 102 upon receipt of an Internet Mobility Protocol message frame from Mobile End System 104. Once the frame is received by the Mobility Management Server 102, the Internet Mobility Protocol engine 244' reconstructs the frame if fragmented (due to the maximum

transmission size of the underlying transport) and then demultiplexes the contents of the message to determine which Mobile End System 104 it was received from. This demultiplexing allows the Internet Mobility Protocol 244' to provide the Remote Procedure Call engine 240' with the correct association- specific context information.

The Internet Mobility Protocol engine 244' then formulates the received message into a RPC receive indication system work request 354, and provides the Mobility Management Server 102 RPC engine 240' with the formulated work request and association-specific context information. When RPC protocol engine 240' receives work request 352, it places it into an association-specific work queue 356, and schedules the association to run by providing a scheduled request to a global queue 358. The main work thread of RPC engine 240' is then signaled that work is available. Once the main thread is awake, it polls the global queue 358 to find the previously queued association scheduled event. It then de-queues the event and beings to process the association-specific work queue 356.

On the association specific work queue 356 it finds the previously queued RPC receive indication work request The main thread then de-queues the RPC receive indication work request 356 and parses the request. Because of the coalescing described in connection with Figure 3A, the Mobility Management Server 102 often receives several RPC transactions bundled in each datagram. It then demultiplexes each RPC transaction back into distinct remote procedure calls and executes the requested function on behalf of Mobile End System 104. For performance purposes RPC engine 240' may provide a look ahead mechanism during the parsing process of the RPC messages to see if it can execute some of the requested transactions concurrently (pipelining).

<u>How RPC Protocol Engine 240' Runs RPC Associations</u>

Figure 4 is a flowchart of an example process for running RPC associations placed on an association work queue 356. When an RPC association is scheduled to run, the main thread for the RPC protocol engine 240' (which may be implemented as a state machine) de-queues the work request from global work queue 358 and determines the type of work request.

There are six basic types of RPC work requests in the preferred embodiment:

- schedule request;
- connect indication
- disconnect indication
- local terminate association
- "resources available" request; and
- ping inactivity timeout.

RPC protocol engine 240' handles these various types of requests differently depending upon its type. RPC protocol engine 240' tests the request type (indicated by information associated with the request as stored on global queue 358) in order to determine how to process the request.

If the type of work request is a "schedule request" (decision block 360), the RPC engine 240' determines which association is being scheduled (block 362). RPC engine 240' can determine this information from what is stored on global queue 358. Once the association is known, RPC engine 240' can identify the particular one of association work queues 356(1) ... 356(n) the corresponding request is stored on. RPC engine 240 retrieves the corresponding association control block (block 362), and calls a Process

Association Work task 364 to begin processing the work in a specific association's work queue 356 as previously noted.

Figure 5 shows example steps performed by the "process association work" task 364 of Figure 4. Once the specific association has been determined, this "process association work" task 364 is called to process the work that resides in the corresponding association work queue 356. If the de-queued work request (block 390) is an RPC receive request (decision block 392), it is sent to the RPC parser to be processed (block 394). Otherwise, if the de-queued work request is a pending receive request (decision block 396), the RPC engine 240' requests TDI 204' to start receiving data on behalf of the application's connection (block 398). If the de-queued work request is a pending connect request (decision block 400), RPC engine 240' requests TDI 204' to issue an application specified TCP (or other transport protocol) connect request (block 402). It then waits for a response from the TDI layer 204'. Once the request is completed by TDI 204', its status is determined and then reported back to the original requesting entity. As a performance measure, RPC engine 240' may decide to retry the connect request process some number of times by placing the request back on the associations-specific work queue (356) before actually reporting an error back to the requesting peer. This again is done in an effort to reduce network bandwidth and processing consumption.

The above process continues to loop until a "scheduling weight complete" test (block 404) is satisfied. In this example, a scheduling weight is used to decide how many work requests will be de-queued and processed for this particular association. This scheduling weight is a configuration parameter set by configuration manager 228, and is acquired when the association connect indication occurs (Figure 4, block 372). This value is configurable based on user or the physical identification of the machine.

Once the RPC engine is finished with the association work queue 356 (for the time at least), it may proceed to process dispatch queues (block 406) (to be discussed in more detail below). If, after processing work on the association's work queue 356, more work remains in the association work queue, the RPC engine 240' will reschedule the association to run again at a later time by posting a new schedule request to the global work queue 358 (Figure 4, decision block 366, block 368; Figure 5, decision block 408, block 410).

Referring once again to Figure 4, if the RPC work request is a "connect indication" (decision block 370), RPC engine 240' is being requested to instantiate a new association with a mobile peer (usually, but not always, the Mobile End System 104). As one example, the connect indication may provide the RPC engine 240' with the following information about the peer machine which is initiating the connection:

- physical identifier of the machine,
- name of the user logged into the machine,
- address of the peer machine, and
- optional connection data from the peer RPC engine 240.

In response to the connect indication (decision block 370), the RPC engine 240 calls the configuration manager 228 with these parameters. Configuration manager 228 uses these parameters to determine the exact configuration for the new connection. The configuration (e.g., association scheduling weight and the list of all applications that require non-default scheduling priorities along with those priorities) is then returned to the RPC engine 240' for storage and execution. RPC engine 240' then starts the new association, and creates a new association control block (block 372). As shown in Figure 5A the following actions may be taken:

- allocate an association control block (block 372A);

- initialize system wide resources with defaults (block 372B);
- get configuration overrides with current configuration settings (block 372C);
- initialize flags (block 372D);
- initialize the association-specific work queue (block 372E);
- initialize association object hash table (block 372F);
- initialize the coalesce timer (block 372G); and
- insert association control block into session table (block 372H).

A "disconnect indication" is issued by the Internet Mobility Protocol engine 244' to the RPC engine 240' when the Internet Mobility Protocol engine has determined that the association must be terminated. The RPC engine 240' tests for this disconnect indication (block 374), and in response, stops the association and destroys the association control block (block 376). As shown in Figure 5B, the following steps may be performed:

- mark the association as deleted to prevent any further processing of work that may be outstanding (block 376A);
- close all associated association objects including process, connection and address objects (block 376B);
- free all elements on work queue (block 376C);
- stop coalesce timer if running (block 376D);
- decrement association control block reference count (block 376E); and
- if the reference count is zero (tested for by block 376F):
- destroy association specific work queue,
- destroy object hash table,
- destroy coalesce timer,
- remove association control block from association table, and

- free control block (376G).

A "terminate session" request is issued when system 102 has determined that the association must be terminated. This request is issued by the system administrator, the operating system or an application. RPC engine 240' handles a terminate session request in the same way it handles a disconnect request (decision block 378, block 376).

In the preferred embodiment, the interface between the RPC engine 240' and the Internet Mobility Protocol engine 244' specifies a flow control mechanism based on credits. Each time one thread posts a work request to another thread, the called thread returns the number of credits left in the work queue. When a queue becomes full, the credit count goes to zero. By convention, the calling thread is to stop posting further work once the credit count goes to zero. Therefore, it is necessary to have a mechanism to tell the calling thread that "resources are available" once the queued work is processed and more room is available by some user configurable/pre-determined low-water mark in the queue. This is the purpose of the "resources available" work indication (tested for by decision block 380). As shown in Figure 5C, the following steps may be performed when the credit count goes to zero:

- mark association as "low mark pending" by setting the RPC_LMPQ_SEND_FLAG (block 379A). Once in this state:
- all received datagrams are discarded (block 379B);
- all received stream events are throttled by refusing to accept the data (block 379C) (this causes the TCP or other transport receive window to eventually close, and provides flow control between the Fixed End System 110 and the Mobility Management Server 102; before returning, the preferred embodiment jams a "pending receive request" to the front of the association specific work

queue 356 so that outstanding stream receive event processing will continue immediately once resources are made available).

- all received connect events are refused for passive connections (block 379D).

When the "resources available" indication is received by the RPC engine 240' (Figure 4, decision block 380), the RPC engine determine whether the association has work pending in its associated association work queue 356; if it does, the RPC engine marks the queue as eligible to run by posting the association to the global work queue 358 (block 382). If a pending receive request has been posted during the time the association was in the low mark pending state, it is processed at this time (in the preferred embodiment, the RPC engine 240 'continues to accept any received connect requests during this processing).

Referring once again to Figure 4, if RPC engine 240' determines that the Mobility Management Server 102 channel used for "ping" has been inactive for a specified period of time (decision block 384), the channel is closed and the resources are freed back to the system to be used by other processes (block 386).

<u>RPC Parsing and Priority Queuing</u>

Referring back to Figure 5, it was noted above that RPC engine parsed an RPC receive request upon receipt (see blocks 392, block 394). Parsing is necessary in the preferred embodiment because a single received datagram can contain multiple RPC calls, and because RPC calls can span multiple Internet Mobility Protocol datagram fragments. An example format for an RPC receive work request 500 is shown in Figure 6. Each RPC receive work request has at least a main fragment 502(1), and may have any number of additional fragments 502(2) .... 502(N). Main fragment

502(1) contains the work request structure header 503 and a receive overlay 504. The receive overlay 504 is a structure overlay placed on top of the fragment 502(1) by the Internet Mobility Protocol engine 244. Within this overlay 504 is a structure member called pUserData that points to the first RPC call 506(1) within the work request 500.

The Figure 6 example illustrates a work request 500 that contains several RPC calls 506(1), 506(2)...506(8). As shown in the Figure 6 example, an RPC work request 500 need not be contained in a contiguous block of memory or in a single fragment 502. In the example shown, a second fragment 502(2) and a third fragment 502(3) that are chained together to the main fragment 502(1) in a linked list.

Thus, RPC parser 394 in this example handles the following boundary conditions:

- each RPC receive request 500 may contain one or more RPC calls;
- one or more RPC calls 506 may exist in a single fragment 502;
- each RPC call 506 may exist completely contained in a fragment 502; and
- each RPC call 506 may span more than one fragment 502.

Figure 7 shows an example RPC parser process 394 to parse an RPC receive work request 500. In this example, the RPC parser 394 gets the first fragment 502(1) in the work request, gets the first RPC call 506(1) in the fragment, and parses that RPC call. Parser 394 proceeds through the RPC receive work request 500 and processes each RPC call 506 in turn. If the number of fragment bytes remaining in the RPC receive work request 500 fragment 502(1) is greater than the size of the RPC header 503, parser 394 determines whether the RPC call is fully contained within the RPC fragment 502 and thus may be processed (this may be determined by testing whether

the RPC call length is greater than the number of fragment bytes remaining). If the RPC call type is a chain exception, then the RPC call will handle the updating of the RPC parser 394 state. In the proxy server 224, the only RPC calls using the chain exception are the "datagram send" and "stream send" calls. This chain exception procedure is done to allow the RPC engine to avoid fragment copies by chaining memory descriptor lists together for the purpose of RPC send calls.

Once the parser 394 identifies an RPC call type, a pointer to the beginning of the RPC information is passed to the RPC engine 240 for execution. The RPC engine divides all TDI procedure calls into different priorities for execution. The highest priority calls are immediately executed by passing them to an RPC dispatcher 395 for immediate execution. All lower priority calls are dispatched to dispatch queues 510 for future processing. Each dispatch queue 510 represents a discrete priority.

In the preferred embodiment, mobile applications call the "open address" object and "open connection" object functions before executing other TDI networking functions. Therefore, the system assigns application level priorities during the "open address" object and "open connection" object calls. In the example embodiment, once an address or connection object is assigned a priority, all calls that are associated with that object are executed within that assigned priority.

If, for example, the RPC call is a TDI Open Address Object request or a TDI Open Connection Object Request, it is sent to the RPC dispatcher 395 for immediate execution. The Open Address and Open Connection object RPC calls provide access to a process ID or process name that are used to match against the information provided by the configuration manager 228 during the configuration requests that occurs within the

association connect indication described earlier. This is used to acquire configuration for the address or connection object.

In the preferred embodiment, all RPC calls have at least an address object or connection object as a parameter. When the call is made, the priority assigned to that specific object is used as the priority for the RPC call. The configuration assigned to the address or connection object determines which priority all associated RPC calls will be executed in. For example, if the assigned priority is "high," all RPC calls will be executed immediately without being dispatched to a dispatch queue 510. If the assigned priority is "1," all RPC calls will be placed into dispatch queue 510(1).

Referring once again to Figure 5, once the "process association work" task 364 process has completed executing its scheduled amount of association work (decision block 404), it checks to see if the dispatch queues require servicing (block 406). Figure 8 is a flowchart of example steps performed by the "process dispatch queues" block 406 of Figure 5 to process the dispatch queues 510 shown in Figure 7.

In this example, dispatch queues 510 are processed beginning with the highest priority queue (510(1) in this example) (block 408). Each queue 510 is assigned a weight factor. The weight factor is a configuration parameter that is returned by the configuration manager 228 when a Mobile End System 104 to Mobility Management Server 102 association is created. As one example, low priority dispatch queues 510 can have a weight factor of 4, and medium priority queues can have a weight factor of 8. High priority RPC calls do not, in this example, use weight factors because they are executed immediately as they are parsed.

RPC engine 240' loops through the de-queuing of RPC calls from the current queue until either the queue is empty or the queue weight

number of RPC calls has been processed (blocks 412-416). For each de-queued RPC call, the RPC dispatcher 395 is called to execute the call. The RPC dispatcher 395 executes the procedural call on behalf of the Mobile End System 104, and formulates the Mobile End System response for those RPC calls that require responses.

If, after exiting the loop, the queue still has work remaining (decision block 418), the queue will be marked as eligible to run again (block 420). By exiting the loop, the system yields the processor to the next lower priority queue (blocks 424, 410). This ensures that all priority levels are given an opportunity to run no matter how much work exists in any particular queue. The system gets the next queue to service, and iterates the process until all queues have been processed. At the end of processing all queues, the system tests to see if any queues have been marked as eligible to run -- and if so, the association is scheduled to run again by posting a schedule request to the global work queue. The association is scheduled to run again in the "process global work" routine shown in Figure 4 above. This approach yields the processor to allow other associations that have work to process an opportunity run. By assigning each queue a weight factor, the system may be tuned to allow different priority levels unequal access to the Mobility Management Server 102's CPU. Thus, higher priority queues are not only executed first, but may also be tuned to allow greater access to the CPU.

<u>Mobility Management Server RPC Responses</u>

The discussion above explains how remote procedure calls are sent from the Mobile End System 104 to the Mobility Management Server 102 for execution. In addition to this type of RPC call, the Mobility Management Server 102 RPC engine 240' also supports RPC events and

RPC receive responses. These are RPC messages that are generated asynchronously as a result of association specific connection peer activity (usually the Fixed End System 110). Mobility Management Server 102 RPC engine 240' completes RPC transactions that are executed by the RPC dispatcher 395. Not all RPC calls require a response on successful completion. Those RPC calls that do require responses on successful completion cause the RPC dispatcher 395 to build the appropriate response and post the response to the Internet Mobile Protocol engine 244' to be retirned to the peer Mobile End System 104. All RPC calls generate a response when the RPC call fails (the RPC receive response is the exception to above).

RPC events originate as a result of network 108 activity by the association specific connection (usually the Fixed End System 110). These RPC event messages are, in the preferred embodiment, proxied by the Mobility Management Server 102 and forwarded to the Mobile End System 104. The preferred embodiment Mobility Management Server 102 supports the following RPC event calls:

- Disconnect Event (this occurs when association-specific connected peer (usually the Fixed End System 110) issues a transport level disconnect request; the disconnect is received by the proxy server 224 on behalf of the Mobile End System 104, and the proxy server then transmits a disconnect event to the Mobile End System);
- Stream Receive Event (this event occurs when the association-specific connected peer (usually the Fixed End System 110) has sent stream data to the Mobile End System 104; the proxy server 224 receives this data on behalf of the Mobile End System 104,

and sends the data to the Mobile End System in the form of a Receive Response);
- Receive Datagram Event (this event occurs when any association-specific portal receives datagrams from a network peer (usually the Fixed End System 110) destined for the Mobile End System 104 through the Mobility Management Server 102; the proxy server 224 accepts these datagrams on behalf of the Mobile End System, and forwards them to the Mobile End System in the form of receive datagram events; and
- Connect Event (this event occurs when the association-specific listening portal receives a transport layer connect request (usually from the Fixed End System 110) when it wishes to establish a transport layer end-to-end connection with a Mobile End System 104; the proxy server 224 accepts the connect request on behalf of the Mobile End System, and then builds a connect event RPC call and forwards it to the Mobile End System).

Figure 9 shows how the RPC engine 240' handles proxy server-generated RPC calls. For high priority address and connection objects, the RPC engine 240' dispatches a send request to the Internet Mobility Protocol engine 244' immediately. The send request results in forwarding the RPC message to the peer Mobile End System 104. For lower priority objects, the Internet Mobility Protocol engine 244 send request is posted to an appropriate priority queue 510'. If the association is not scheduled to run, a schedule request is also posted to the global queue 358'. The Internet Mobility Protocol send request is finally executed when the dispatch queues are processed as described earlier in connection with Figures 5 & 8.

Example Internet Mobility Protocol

Internet Mobility Protocol provided in accordance with the present invention is a message oriented connection based protocol. It provides guaranteed delivery, (re)order detection, and loss recovery. Further, unlike other conventional connection oriented protocols (i.e. TCP), it allows for multiple distinct streams of data to be combined over a single channel; and allows for guaranteed, unreliable, as well as new message oriented reliable data to traverse the network through the single virtual channel simultaneously. This new message oriented level of service can alert the requester when the Internet Mobility Protocol peer has acknowledged a given program data unit.

The Internet Mobility Protocol provided in accordance with the present invention is designed to be an overlay on existing network topologies and technologies. Due to its indifference to the underlying network architecture, it is transport agnostic. As long as there is a way for packetized data to traverse between two peers, Internet Mobility Protocol can be deployed. Each node's network point of presence (POP) or network infrastructure can also be changed without affecting the flow of data except where physical boundary, policy or limitations of bandwidth apply.

With the help of the layer above, Internet Mobility Protocol coalesces data from many sources and shuttles the data between the peers using underlying datagram facilities. As each discrete unit of data is presented from the upper layer, Internet Mobility Protocol combines into a single stream and subsequently submits it for transmission. The data units are then forwarded to the peer over the existing network where upon reception, with the help from the layer above, the stream is demultiplexed back into multiple distinct data units. This allows for optimum use of available bandwidth, by generating the maximum sized network frames

possible for each new transmission. This also has the added benefit of training the channel once for maximum bandwidth utilization and have its parameters applied to all session level connections.

In rare instances where one channel is insufficient, the Internet Mobility Protocol further allows multiple channels to be established between the peers -- thus allowing for data prioritization and possibly providing a guaranteed quality of service (if the underlying network provides the service).

The Internet Mobility Protocol also provides for dynamically selectable guaranteed or unreliable levels of service. For example, each protocol data unit that is submitted for transmission can be queued with either a validity time period or a number of retransmit attempts or both. Internet Mobility Protocol will expire a data unit when either threshold is reached, and remove it from subsequent transmission attempts.

Internet Mobility Protocol's additional protocol overhead is kept minimal by use of variable length header. The frame type and any optional fields determine the size of the header. These optional fields are added in a specific order to enable easy parsing by the receiving side and bits in the header flag field denote their presence. All other control and configuration information necessary for the peers to communicate can be passed through the in-band control channel. Any control information that needs to be sent is added to the frame prior to any application level protocol data unit. The receiving side processes the control information and then passes the rest of the payload to the upper layer.

Designed to run over relatively unreliable network links where the error probability is relatively high, Internet Mobility Protocol utilizes a number of techniques to insure data integrity and obtain optimum network performance. To insure data integrity, a Fletcher checksum algorithm is

used to detect errant frames. This algorithm was selected due to the fact of its efficiency as well as its detection capability. It can determine not only bit errors, but also bit reordering. However, other alternate checksum algorithms maybe used in its place.

Sequence numbers are used to insure ordered delivery of data. Internet Mobility Protocol sequence numbers do not, however, represent each byte of data as in TCP. They represent a frame of data that can be, in one example implementation, as large as 65535 bytes (including the Internet Mobility Protocol header). They are 32 bits or other convenient length in one example to insure that wrap-around does not occur over high bandwidth links in a limited amount of time.

Combining this capability along with the expiration of data, retransmitted (retried) frames may contain less information than the previous version that was generated by the transmitting side. A frame id is provided to enable detection of the latest versioned frame. However, since data is never added in the preferred embodiment and each element removed is an entire protocol data unit, this is not a necessity for sequence assurance. In one example, the Internet Mobility Protocol will only process the first instance of a specific frame it receives -- no matter how many other versions of that frame are transmitted. Each frame created that carries new user payload is assigned its own unique sequence number.

Performance is gained by using of a sliding window technique -- thus allowing for more then one frame to be outstanding (transmitted) at a time before requiring the peer to acknowledge reception of the data. To insure timely delivery of the data, a positive acknowledgement and timer based retransmit scheme is used. To further optimize the use of the channel, a selective acknowledgement mechanism is employed that allows for fast retransmission of missing frames and quick recovery during lossy or

congested periods of network connectivity. In one example, this selective acknowledgement mechanism is represented by an optional bit field that is included in the header.

A congestion avoidance algorithm is also included to allow the protocol to back off from rapid retransmission of frames. For example, a round trip time can be calculated for each frame that has successfully transfer between the peers without a retransmit. This time value is averaged and then used as the basis for the retransmission timeout value. As each frame is sent, a timeout is established for that frame. If an acknowledgement for that frame is not received, and the frame has actually been transmitted, the frame is resent. The timeout value is then increased and then used as the basis for the next retransmission time. This retransmit time-out is bounded on both the upper and lower side to insure that the value is within a reasonable range.

Internet Mobility Protocol also considers the send and receive paths separately. This is especially useful on channels that are asymmetric in nature. Base on hysteresis, the Internet Mobility Protocol automatically adjusts parameters such as frame size (fragmentation threshold), number of frames outstanding, retransmit time, and delayed acknowledgement time to reduce the amount of duplicate data sent through the network.

Due to the fact that Internet Mobility Protocol allows a node to migrate to different points of attachment on diverse networks, characteristics (e.g., frame size) of the underlying network may change midstream. An artifact of this migration is that frames that have been queued for transmission on one network may no longer fit over the new medium the mobile device is currently attached to. Combining this issue with the fact that fragmentation may not be supported by all network infrastructures, fragmentation is dealt with at the Internet Mobility Protocol

level. Before each frame is submitted for transmission, Internet Mobility Protocol assesses whether or not it exceeds the current fragmentation threshold. Note that this value may be less than the current maximum transmission unit for performance reason (smaller frames have a greater likelihood of reaching its ultimate destination then larger frames). The tradeoff between greater protocol overhead versus more retransmissions is weighed by Internet Mobility Protocol, and the frame size may be reduced in an attempt to reduce overall retransmissions). If a given frame will fit, it is sent in its entirety. If not, the frame is split into maximum allowable size for the given connection. If the frame is retransmitted, it is reassessed, and will be refragmented if the maximum transmission unit has been reduced (or alternatively, if the maximum transmission unit actually grew, the frame may be resent as a single frame without fragmentation).

The protocol itself is orthogonal in its design as either side may establish or terminate a connection to its peer. In a particular implementation, however, there may be a few minor operational differences in the protocol engine depending on where it is running. For example, based on where the protocol engine is running, certain inactivity detection and connection lifetime timeouts may be only invoked on one side. To allow administrative control, Internet Mobility Protocol engine running on the Mobility Management Server 102 keeps track of inactivity periods. If the specified period of time expires without any activity from the Mobile End System 104, the Mobility Management Server 102 may terminate a session. Also, an administrator may want to limit the overall time a particular connection may be established for, or when to deny access base on time of day. Again these policy timers may, in one example implementation, be invoked only on the Mobility Management Server 102 side.

In one example implementation, the software providing the Internet Mobility Protocol is compiled and executable under Windows NT, 9x, and CE environments with no platform specific modification. To accomplish this, Internet Mobility Protocol employs the services of a network abstraction layer (NAL) to send and receive Internet Mobility Protocol frames. Other standard utility functions such as memory management, queue and list management, event logging, alert system, power management, security, etc are also used. A few runtime parameters are modified depending on whether the engine is part of a Mobile End System 104 or Mobility Management Server 102 system. Some examples of this are:

- Certain timeouts are only invoked on the Mobility Management Server 102
- Direction of frames are indicated within each frame header for echo detection
- Inbound connections may be denied if Mobile End System 104 is so configured
- Alerts only signaled on Mobility Management Server 102
- Power management enabled on Mobile End System 104 but is not necessary on the Mobility Management Server 102

The Internet Mobility Protocol interface may have only a small number of "C" callable platform independent published API functions, and requires one O/S specific function to schedule its work (other then the aforementioned standard utility functions). Communications with local clients is achieved through the use of defined work objects (work requests). Efficient notification of the completion of each work element is accomplished by signaling the requesting entity through the optional completion callback routine specified as part of the work object.

The Internet Mobility Protocol engine itself is queue based. Work elements passed from local clients are placed on a global work queue in FIFO order. This is accomplished by local clients calling a published Internet Mobility protocol function such as "ProtocolRequestwork()". A scheduling function inside of Internet Mobility Protocol then removes the work and dispatches it to the appropriate function. Combining the queuing and scheduling mechanisms conceal the differences between operating system architectures -- allowing the protocol engine to be run under a threaded based scheme (e.g., Windows NT) or in a synchronous fashion (e.g., Microsoft Windows 9x & Windows CE). A priority scheme can be overlaid on top of its queuing, thus enabling a guaranteed quality of service to be provided (if the underlying network supports it).

From the network perspective, the Internet Mobility Protocol uses scatter-gather techniques to reduce copying or movement of data. Each transmission is sent to the NAL as a list of fragments, and is coalesced by the network layer transport. If the transport protocol itself supports scatter-gather, the fragment list is passed through the transport and assembled by the media access layer driver or hardware. Furthermore, this technique is extensible in that it allows the insertion or deletion of any protocol wrapper at any level of the protocol stack. Reception of a frame is signaled by the NAL layer by calling back Internet Mobility Protocol at a specified entry point that is designated during the NAL registration process.

Example Internet Mobility Protocol Engine Entry Points

Internet Mobility Protocol in the example embodiment exposes four common entry points that control its startup and shutdown behavior. These procedures are:

1. Internet Mobility ProtocolCreate()

2. Internet Mobility ProtocolRun()
3. Internet Mobility ProtocolHalt()
4. Internet Mobility ProtocolUnload()

Example Internet Mobility ProtocolCreate()

The Internet Mobility ProtocolCreate() function is called by the boot subsystem to initialize the Internet Mobility Protocol. During this first phase, all resource necessary to start processing work must be acquired and initialized. At the completion of this phase, the engine must be in a state ready to accept work from other layers of the system. At this point, Internet Mobility Protocol initializes a global configuration table. To do this, it employs the services of the Configuration Manager 228 to populate the table.

Next it registers its suspend and resume notification functions with the APM handler. In one example, these functions are only invoked on the Mobile End System 104 side -- but in another implementation it might be desirable to allow Mobility Management Server 102 to suspend during operations. Other working storage is then allocated from the memory pool, such as the global work queue, and the global NAL portal list.

To limit the maximum amount of runtime memory required as well as insuring Internet Mobility Protocol handles are unique, Internet Mobility Protocol utilizes a 2-tier array scheme for generating handles. The globalConnectionArray table is sized based on the maximum number of simultaneous connection the system is configured for, and allocated at this time. Once all global storage is allocated and initialized, the global Internet Mobility Protocol state is change to _STATE_INITIALIZE_.

Example Internet Mobility ProtocolRun()

The Internet Mobility ProtocolRun() function is called after all subsystems have been initialized, and to alert the Internet Mobility Protocol subsystem that it is okay to start processing any queued work. This is the normal state that the Internet Mobility Protocol engine is during general operations. A few second pass initialization steps are taken at this point before placing the engine into an operational state.

Internet Mobility Protocol allows for network communications to occur over any arbitrary interface(s). During the initialization step, the storage for the interface between Internet Mobility Protocol and NAL was allocated. Internet Mobility Protocol now walks through the global portal list to start all listeners at the NAL. In one example, this is comprised of a two step process:

- Internet Mobility Protocol requests the NAL layer to bind and open the portal based on configuration supplied during initialization time; and
- Internet Mobility Protocol then notifies the NAL layer that it is ready to start processing received frames by registering the Internet Mobility ProtocolRCVFROMCB call back.
- A local persistent identifier (PID) is then initialized.

The global Internet Mobility Protocol state is change to _STATE_RUN_.

Example Internet Mobility ProtocolHalt

The Internet Mobility ProtocolHalt() function is called to alert the engine that the system is shutting down. All resources acquired during its operation are to be release prior to returning from this function. All Internet Mobility Protocol sessions are abnormally terminated with the reason code

set to administrative. No further work is accepted from or posted to other layers once the engine has entered into _STATE_HALTED_ state.

Example Internet Mobility ProtocolUnload()

The Internet Mobility ProtocolUnload() function is the second phase of the shutdown process. This is a last chance for engine to release any allocated system resources still being held before returning. Once the engine has returned from this function, no further work will be executed as the system itself is terminating

Example Internet Mobility Protocol handles

In at least some examples, using just the address of the memory (which contains the Internet Mobility Protocol state information) as the token to describe an Internet Mobility Protocol connection may be insufficient. This is mainly due to possibility of one connection terminating and a new one starting in a short period of time. The probability that the memory allocator will reassign the same address for different connections is high -- and this value would then denote both the old connection and a new connection. If the original peer did not hear the termination of the session (i.e. it was off, suspended, out of range, etc.), it could possibly send a frame on the old session to the new connection. This happens in TCP and will cause a reset to be generated to the new session if the peer's IP addresses are the same. To avoid this scenario, Internet Mobility Protocol uses manufactured handle. The handles are made up of indexes into two arrays and a nonce for uniqueness. The tables are laid out as follows.

Table 1: an array of pointers to an array of connection object

Table 2 : an array of connection objects that contains the real pointers to the Internet Mobility Protocol control blocks.

This technique minimizes the amount of memory being allocated at initialization time. Table 1 is sized and allocated at startup. On the Mobile End System 104 side this allows allocation of a small amount of memory (the memory allocation required for this Table 1 on the Mobility Management Server 102 side is somewhat larger since the server can have many connections).

Table 1 is then populated on demand. When a connection request is issued, Internet Mobility Protocol searches through Table 1 to find a valid pointer to Table 2. If no entries are found, then Internet Mobility Protocol will allocate a new Table 2 with a maximum of 256 connection objects -- and then stores the pointer to Table 2 into the appropriate slot in Table 1. The protocol engine then initializes Table 2, allocates a connection object from the newly created table, and returns the manufactured handle. If another session is requested, Internet Mobility Protocol will search Table 1 once again, find the valid pointer to Table 2, and allocate the next connection object for the session. This goes on until one of two situations exist:

- If all the connection objects are exhausted in Table 2, a new Table 2 will be allocated, initialized, and a pointer to it will be placed in the next available slot in Table 1; and
- If all connection objects have been released for a specific Table 2 instance and all elements are unused for a specified period of time, the storage for that instance of Table 2 is released back to the memory pool and the associated pointer in Table 1 is zeroed to indicate that that entry is now available for use when the next connection request is started (if and only if no other connection object are available in other instances of Table 2).

Two global counters are maintained to allow limiting the total number of connections allocated. One global counter counts the number of current active connections; and the other keeps track of the number of unallocated connection objects. The second counter is used to govern the total number of connection object that can be created to some arbitrary limit. When a new Table 2 is allocated, this counter is adjusted downward to account for the number of objects the newly allocated table represents. On the flip side, when Internet Mobility Protocol releases a Table 2 instance back to the memory pool, the counter is adjusted upward with the number of connection objects that are being released.

Example Work Flow

Work is requested by local clients through the Internet Mobility ProtocolRequestWork() function. Once the work is validated and placed on the global work queue, the Internet Mobility ProtocolWorkQueueEligible() function is invoked. If in a threaded environment, the Internet Mobility Protocol worker thread is signaled (marked eligible) and control is immediately returned to the calling entity. If in a synchronous environment, the global work queue is immediately run to process any work that was requested. Both methods end up executing the Internet Mobility ProtocolProcessWork() function. This is the main dispatching function for processing work.

Since only one thread at a time may be dispatching work from the global queue in the example embodiment, a global semaphore may be used to protect against reentrancy. Private Internet Mobility Protocol work can post work directly to the global work queue instead of using the Internet Mobility ProtocolRequestWork() function.

A special case exists for SEND type work objects. To insure that the semantics of Unreliable Datagrams is kept, each SEND type work object can be queued with an expiry time or with a retry count. Work will be aged based on the expiry time. If the specified timeout occurs, the work object is removed from the connection specific queue, and is completed with an error status. If the SEND object has already been coalesced into the data path, the protocol allows for the removal of any SEND object that has specified a retry count. Once the retry count has been exceeded, the object is removed from the list of elements that make up the specific frame, and then returned to the requestor with the appropriate error status.

Example Connection Startup

Internet Mobility Protocol includes a very efficient mechanism to establish connections between peers. Confirmation of a connection can be determined in as little as a three-frame exchange between peers. The initiator sends an IMP SYNC frame to alert its peer that it is requesting the establishment of a connection. The acceptor will either send an IMP ESTABLISH frame to confirm acceptance of the connection, or send an IMP ABORT frame to alert the peer that its connection request has been rejected. Reason and status codes are passed in the IMP ABORT frame to aid the user in deciphering the reason for the rejection. If the connection was accepted, an acknowledgement frame is sent (possibly including protocol data unit or control data) and is forwarded to the acceptor to acknowledge receipt of its establish frame.

To further minimize network traffic, the protocol allows user and control data to be included in the initial handshake mechanism used at connection startup. This ability can be used in an insecure environment or in environments where security is dealt with by a layer below, such that the

Internet Mobility Protocol can be tailored to avert the performance penalties due to double security authentication and encryption processing being done over the same data path.

Example Data transfer

Internet Mobility Protocol relies on signaling from the NAL to detect when a frame has been delivered to the network. It uses this metric to determine if the network link in question has been momentarily flow controlled, and will not submit the same frame for retransmission until the original request has been completed. Some network drivers however lie about the transmission of frames and indicate delivery prior to submitting them to the network. Through the use of semaphores, the Internet Mobility Protocol layer detects this behavior and only will send another datagram until the NAL returns from the original send request

Once a frame is received by Internet Mobility Protocol, the frame is quickly validated, then placed on an appropriate connection queue. If the frame does not contain enough information for Internet Mobility Protocol to discern its ultimate destination, the frame is placed on the Internet Mobility Protocol socket queue that the frame was received on, and then that socket queue is place on the global work queue for subsequence processing. This initial demultiplexing allows received work to be dispersed rapidly with limited processing overhead.

Example Acquiescing

To insure minimal use of network bandwidth during periods of retransmission and processing power on the Mobility Management Server 102, the protocol allows the Mobility Management Server 102 to "acquiesce" a connection. After a user configurable period of time, the

Mobility Management Server 102 will stop retransmitting frames for a particular connection if it receives no notification from the corresponding Mobile End System 104. At this point, the Mobility Management Server 102 assumes that the Mobile End System 104 is in some unreachable state (i.e. out of range, suspended, etc), and places the connection into a dormant state. Any further work destined for this particular connection is stored for future delivery. The connection will remain in this state until one of the following conditions are met:

- Mobility Management Server 102 receives a frame from the Mobile End System 104, thus returning the connection to its original state;
- a lifetime timeout has expired;
- an inactivity timeout has expired; or
- the connection is aborted by the system administrator.

In the case that the Mobility Management Server 102 receives a frame from the Mobile End System 104, the connection continues from the point it was interrupted. Any work that was queued for the specific connection will be forwarded, and the state will be resynchronized. In any of the other cases, the Mobile End System 104 will be apprised of the termination of the connection once it reconnects; and work that was queued for the Mobile End System 104 will be discarded.

<u>Example Connect and Send Requests</u>

Figures 10A-10C together are a flowchart of example connect and send request logic formed by Internet mobility engine 244. In response to receipt from a command from RPC engine 240, the Internet Mobility Protocol engine 244 determines whether the command is a "connect" request (decision block 602). If it is, engine 244 determines whether

connection resources can be allocated (decision block 603). If it is not possible to allocate sufficient connection resources ("no" exit to decision block 603), engine 244 declares an error (block 603a) and returns. Otherwise, engine 244 performs a state configuration process in preparation for handling the connect request (block 603b).

For connect and other requests, engine 244 queues the connect or send request and signals a global event before return to the calling application (block 604).

To dispatch a connect or send request from the Internet Mobility Protocol global request queue, engine 244 first determines whether any work is pending (decision block 605). If no work is pending ("no" exit to decision block 605), engine 244 waits for the application to queue work for the connection by going to Figure 10C, block 625 (block 605a). If there is work pending ("yes" exit to decision block 605), engine 244 determines whether the current state has been established (block 606). If the state establish has been achieved ("yes" exit to decision block 606), engine 244 can skip steps used to transition into establish state and jump to decision block 615 of Figure 10B (block 606a). Otherwise, engine 244 must perform a sequence of steps to enter establish state ("no" exit to decision block 606).

In order to enter establish state, engine 244 first determines whether the address of its peer is known (decision block 607). If not, engine 244 waits for the peer address while continuing to queue work and transitions to Figure 10C block 625 (block 607a). If the peer address is known ("yes" exit to decision block 607), engine 244 next tests whether the requisite security context has been acquired (decision block 608). If not, engine 244 must wait for the security context while continuing to queue work and transitioning to block 625 (block 608a). If security context has already been acquired ("yes" exit to decision block 608), engine 244 declares a "state

pending" state (block 608b), and then sends an Internet Mobility Protocol sync frame (block 609) and starts a retransmit timer (block 610). Engine 244 determines whether the corresponding established frame was received (block 611). If it was not ("no" exit to decision block 611), engine 244 tests whether the retransmit time has expired (decision block 612). If the decision block has not expired ("no" exit to decision block 612), engine 244 waits and may go to step 625 (block 613). Eventually, if the established frame is never received (as tested for by block 611) and a total retransmit time expires (decision block 614), the connection may be aborted (block 614a). If the established is eventually received ("yes" exit to decision block 611), engine 244 declares a "state established" state (block 611a).

Once state establish has been achieved, engine 244 tests whether the new connection has been authenticated (decision block 615). If it has not been, engine 244 may wait and transition to step 625 (block 616). If the connection has been authenticated ("yes" exit to decision block 615), engine 244 tests whether authentication succeeded (decision block 617). If it did not ("no" exit to decision block 617), the connection is aborted (block 614a). Otherwise, engine 244 tests whether the peer transmit window is full (decision block 618). If it is ("yes" exit to decision block 618), engine 244 waits for acknowledgment and goes to step 625 (decision block 619). If the window is not full ("no" exit to decision block 618), engine 244 creates an Internet Mobility Protocol data frame (block 620) and sends it (block 621). Engine 244 then determines if the retransmit timer has started (decision block 622). If no, engine 244 starts the retransmit timer (block 623). Engine 244 loops through blocks 618-623 until there is no more data to send (as tested for by decision block 624). Engine 244 then returns to a sleep mode waiting for more work and returns to the global dispatcher (block 625).

Example Termination

Figure 11 is a flowchart of example steps performed by Internet Mobility Protocol engine 244 to terminate a connection. In response to a "terminate connection" request (block 626), the engine queues the request to its global work queue and returns to the calling application (block 626a). The terminate request is eventually dispatched from the Internet Mobility Protocol process global work queue for execution (block 627). Engine 244 examines the terminate request and determines whether the terminate request should be immediate or graceful (decision block 628). If immediate ("abort" exit to decision block 628), engine 244 immediately aborts the connection (block 629). If graceful ("graceful" exit to decision block 628), engine 244 declares a "state close" state (block 628a), and sends an Internet Mobility Protocol "Mortis" frame (block 630) to indicate to the peer that the connection is to close. Engine 244 then declares a "Mortis" state (block 630a) and starts the retransmit timer (block 631). Engine 244 tests whether the response of "post mortem" frame has been received from the peer (decision block 632). If not ("no" exit to decision block 632), engine 244 determines whether a retransmit timer has yet expired (decision block 633). If the retransmit timer is not expired ("no" exit to decision block 633), engine 244 waits and proceeds to step 637 (block 634). If the retransmit timer has expired ("yes" exit to decision block 633), engine 244 determines whether the total retransmit time has expired (decision block 635). If the total time is not yet expired ("no" exit to decision block 635), control returns to block 630 to resent the Mortis frame. If the total retransmit time has expired ("yes" exit to decision block 635), engine 244 immediately aborts the connection (block 635a).

Once a "post mortem" responsive frame has been received from the peer ("yes" exit to decision block 632), engine 244 declares a "post mortem"

state (block 632a), releases connection resources (block 636), and returns to sleep waiting for more work (block 637).

Example Retransmission

Figure 12 is a flowchart of example "retransmit" events logic performed by Internet Mobility Protocol engine 244. In the event that the retransmit timer has expired (block 650), engine 244 determines whether any frames are outstanding (decision block 651). If no frames are outstanding ("no" exit to decision block 651), engine 244 dismisses the timer (block 652) and returns to sleep (block 660). If, on the other hand, frames are outstanding ("yes" exit to decision block 651), engine 244 determines whether the entire retransmit period has expired (decision block 653). If it has not ("no" exit to decision block 653), the process returns to sleep for the difference in time (block 654). If the entire retransmit time period has expired ("yes" exit to decision block 653), engine 244 determines whether a total retransmit period has expired (decision block 655). If it has ("yes" exit to decision block 655) and this event has occurred in the Mobility Management Server engine 244' (as opposed to the Mobile End System engine 244), a dormant state is declared (decision block 656, block 656a). Under these same conditions, the Internet Mobility Protocol engine 244 executing on the Mobile End System 104 will abort the connection (block 656b).

If the total retransmit period is not yet expired ("no" exit to decision block 655), engine 244 reprocesses the frame to remove any expired data (block 657) and then retransmits it (block 658) -- restarting the retransmit timer as it does so (block 659). The process then returns to sleep (block 660) to wait for the next event.

Example Internet Mobility Protocol expiration of a PDU

Figure 12 block 657 allows for the requesting upper layer interface to specify a timeout or retry count for expiration of any protocol data unit (i.e. a SEND work request) submitted for transmission to the associated peer. By use of this functionality, Internet Mobility Protocol engine 244 maintains the semantics of unreliable data and provides other capabilities such as unreliable data removal from retransmitted frames. Each PDU (protocol data unit) 506 submitted by the layer above can specify a validity timeout and/or retry count for each individual element that will eventually be coalesced by the Internet Mobility Protocol engine 244. The validity timeout and/or retry count (which can be user-specified for some applications) are used to determine which PDUs 506 should not be retransmitted but should instead be removed from a frame prior to retransmission by engine 244.

The validity period associated with a PDU 506 specifies the relative time period that the respective PDU should be considered for transmission. During submission, the Internet Mobility Protocol RequestWork function checks the expiry timeout value. If it is non-zero, an age timer is initialized. The requested data is then queued on the same queue as all other data being forwarded to the associated peer. If the given PDU 506 remains on the queue for longer than the time period specified by the validity period parameter, during the next event that the queue is processed, the given (all) PDU(s) that has an expired timeout is removed and completed locally with a status code of "timeout failure" rather than being retransmitted when the frame is next retransmitted. This algorithm ensures that unreliable data being queued for transmission to the peer will not grow stale and/or boundlessly consume system resources.

In the example shown in Figure 12A, at least three separate PDUs 506 are queued to Internet Mobility Protocol engine 244 for subsequent processing. PDU 506(1) is queued without an expiry time denoting no timeout for the given request. PDU 506(2) is specified with a validity period of 2 seconds and is chronologically queued after PDU 506(1). PDU 506(n) is queued 2.5 seconds after PDU 506(2) was queued. Since the act of queuing PDU 506(n) is the first event causing processing of the queue and PDU 506(2) expiry time has lapsed, PDU 506(2) is removed from the work queue, completed locally and then PDU 506(n), is placed on the list. If a validity period was specified for PDU 506(n) the previous sequence of events would be repeated. Any event (queuing, dequeuing, etc) that manipulates the work queue will cause stale PDUs to be removed and completed.

As described above, PDUs 506 are coalesced by the Internet Mobility Protocol Engine 244 transmit logic and formatted into a single data stream. Each discrete work element, if not previously expired by the validity timeout, is gathered to formulate Internet Mobility Protocol data frames. Internet Mobility Protocol Engine 244 ultimately sends these PDUs 506 to the peer, and then places the associated frame on a Frames-Outstanding list. If the peer does not acknowledge the respective frame in a predetermined amount of time (see Figure 12 showing the retransmission algorithm), the frame is retransmitted to recover from possibly a lost or corrupted packet exchange. Just prior to retransmission, the PDU list that the frame is comprised of is iterated through to determine if any requests were queued with a retry count. If the retry count is non zero, and the value is decremented to zero, the PDU 506 is removed from the list, and the frames header is adjusted to denote the deletion of data. In this fashion, stale data, unreliable data, or applications employing their own

retransmission policy are not burdened by engine 244's retransmission algorithm.

In the Figure 12B example, again three separate PDUs 506 are queued to Internet Mobility Protocol engine 244 for subsequent processing. PDU 506(1) is queued without a retry count. This denotes continuous retransmission attempts or guaranteed delivery level of service. PDU 506(2) is queued with a retry count of 1 and is chronologically queued after PDU 506(1). PDU 506(n) is queued sometime after PDU 506(2). At this point, some external event (e.g., upper layer coalesce timer, etc.) causes engine 244's send logic to generate a new frame by gathering enough PDUs 506 from the work queue to generate an Internet Mobility Protocol data frame 500. The frame header 503 is calculated and stamped with a retry ID of 0 to denote that this is the first transmission of the frame. The frame is then handed to the NAL layer for subsequent transmission to the network. At this point a retransmit timer is started since the frame in question contains a payload. For illustration purposes it is assumed that an acknowledgement is not received from the peer for a variety of possible reasons before the retransmit timer expires. The retransmit logic of engine 244 determines that the frame 500 in question is now eligible for retransmission to the network. Prior to resubmitting the frame to the NAL layer, engine 244's retransmit logic iterates through the associated list of PDUs 506. Each PDU's retry count is examined and if non-zero, the count is decremented. In the process of decrementing PDU 506(2)'s retry count, the retry count becomes zero. Because PDU 506(2)'s retry count has gone to zero, it is removed from the list and completed locally with a status of "retry failure." The frame header 503 size is then adjusted to denote the absence of the PDU 506(2)'s data. This process is repeated for all remaining PDUs. Once the entire frame 500 is reprocessed to produce an

"edited" frame 500', the retry ID in the header is incremented and the resultant datagram is then handed to the NAL layer for subsequent (re)transmission.

Example Reception

Figures 13A-13D are a flowchart of example steps performed by Internet Mobility Protocol engine 244 in response to receipt of a "receive" event. Such receive events are generated when an Internet Mobility Protocol frame has been received from network 108. In response to this receive event, engine 244 pre-validates the event (block 670) and tests whether it is a possible Internet Mobility Protocol frame (decision block 671). If engine 244 determines that the received frame is not a possible frame ("no" exit to decision block 671), it discards the frame (block 672). Otherwise ("yes" exit to decision block 671), engine 244 determines whether there is a connection associated with the received frame (decision block 673). If there is a connection associated with the received frame ("yes" exit to decision block 673), engine 244 places the work on the connection receive queue (block 674), marks the connection as eligible to receive (block 675), and places the connection on the global work queue (block 676). If no connection has yet been associated with the received frame ("no" exit to decision block 673), engine 244 places the received frame on the socket receive queue (block 677) and places the socket receive queue on the global work queue (block 678). In either case, engine 244 signals a global work event (block 679). Upon dispatching of a "receive eligible" event from the global work queue (see Figure 13B), engine 244 de-queues the frame from the respective receive queue (block 680). It is possible that more then one IMP frame is received and queued before the Internet Mobility Protocol engine 244 can start de-queuing the messages.

Engine 244 loops until all frames have been de-queue (blocks 681, 682). Once a frame has been de-queued ("yes" exit to decision block 681), engine 244 validates the received frame (block 683) and determines whether it is okay (decision block 684). If the received frame is invalid, engine 244 discards it (block 685) and de-queues the next frame from the receive queue (block 680). If the received frame is valid ("yes" exit to decision block 684), engine 244 determines whether it is associated with an existing connection (block 686). If it is not ("no" exit to decision block 686), engine 244 tests whether it is a sync frame (decision block 687). If it is not a sync frame ("no" exit to decision block 687), the frame is discarded (block 685). If, on the other hand, a sync frame has been received ("yes" exit to decision block 687), engine 244 processes it using a passive connection request discussed in association with Figures 14A and 14B (block 688).

If the frame is associated with a connection ("yes" exit to decision block 686), engine 244 determines whether the connection state is still active and not "post mortem" (decision block 689). If the connection is already "post mortem," the frame is discarded (block 685). Otherwise, engine 244 parses the frame (block 690) and determines whether it is an abort frame (decision block 691). If the frame is an abort frame, engine 244 immediately aborts the connection (block 691a). If the frame is not an abort frame ("yes" exit to decision block 691), engine 244 processes acknowledgment information and releases any outstanding send frames (block 692). Engine 244 then posts the frame to any security subsystem for possible decryption (block 693). Once the frame is returned from the security subsystem engine 244 processes any control data (block 694). Engine 244 then determines whether the frame contains application data (decision block 695). If it does, this data is queued to the application layer (block 696). Engine 244 also determines whether the connection's state is

dormant (block 697 and 697a -- this can happen on Mobility Management Server engine 244' in the preferred embodiment), and returns state back to established.

If the frame is possibly a "Mortis" frame ("yes" exit to decision block 698), engine 244 indicates a "disconnect" to the application layer (block 699) and enters the "Mortis" state (block 699a). It sends a "post mortem" frame to the peer (block 700), and enters the "post mortem" state (block 700a). Engine 244 then releases connection resources (block 701) and returns to sleep waiting for more work (block 702). If the parsed frame is a "post mortem" frame ("yes" exit to decision block 703), blocks 700a, 701, 702 are executed. Otherwise, control returns to block 680 to dequeue the next frame from the receive queue (block 704).

Example Passive Connections

Blocks 14A-14B are together a flowchart of example steps performed by Internet Mobility Protocol engine 244 in response to a "passive connection" request. Engine 244 first determines whether there is another connection for this particular device (block 720). If there is ("yes" exit to decision block 720), the engine determines whether it is the initial connection (decision block 721). If peer believes the new connection is the initial connection ("yes" exit to decision block 721), engine 244 aborts the previous connections (block 722). If not the initial connection ("no" exit to decision block 721), engine 244 tests whether the sequence and connection ID match (decision block 723). If they do not match ("no" exit to decision block 723), control returns to decision block 720. If the sequence and connection ID do match ("yes" exit to decision block 723), engine 244 discards duplicate frames (block 724) and returns to step 680 of Figure 13B (block 725).

If there is no other connection ("no" exit to decision block 720), engine 244 determines whether it can allocate connection resources for the connection (decision block 726). If it cannot, an error is declared ("no" exit to decision block 726, block 727), and the connection is aborted (block 728). If it is possible to allocate connection resources ("yes" exit to decision block 726), engine 244 declares a "configure" state (block 726a) and acquires the security context for the connection (block 730). If it was not possible to acquire sufficient security context ("no" exit to decision block 731), the connection is aborted (block 728). Otherwise, engine 244 sends an established frame (block 732) and declares the connection to be in state "establish" (block 732a). Engine 244 then starts a retransmitter (block 733) and waits for the authentication process to conclude (block 734). Eventually, engine 244 tests whether the device and user have both been authenticated (block 735). If either the device or the user is not authenticated, the connection is aborted (block 736). Otherwise, engine 244 indicates the connection to the listening application (block 737) and gets the configuration (block 738). If either of these steps do not succeed, the connection is aborted (decision block 739, block 740). Otherwise, the process returns to sleep waiting for more work (block 741).

Example Abnormal Termination

Figures 15A and 15B are a flowchart of example steps performed by the Internet Mobility Protocol engine 244 in response to an "abort" connection request. Upon receipt of such a request from another process (block 999) and dispatched via the queue (block 1000), engine 244 determines whether a connection is associated with the request (decision block 1001). If it is ("yes" exit to decision block 1001), engine 244 saves the original state (block 1002) and declares an "abort" state (block 1002a).

Engine 244 then determines whether the connection was indicated to the RPC engine (decision block 1003) – and if so, indicates a disconnect event(block 1004). Engine 244 then declares a "post mortem" state (block 1003a), releases the resources previously allocated to the particular connection (block 1005), and tests whether the original state is greater than the state pending (decision block 1006). If not ("no" exit to decision block 1006), the process transitions to block 1012 to return to the calling routine (block 1007). Otherwise, engine 244 determines whether the request is associated with a received frame (decision block 1008). If the abort request is associated with a received frame, and the received frame is an abort frame (decision block 1009), the received frame is discarded (block 1010). Otherwise engine 244 will send an abort frame (block 1011) before returning to the calling routine (block 1012).

Example Roaming Control

Referring once again to Figure 1, mobile network 108 may comprise a number of different segments providing different network interconnects (107a-107k corresponding to different wireless transceivers 106a-106k). In accordance with another aspect of the present invention, network 108 including Mobility Management Server 102 is able to gracefully handle a "roaming" condition in which a Mobile End System 104 has moved from one network interconnect to another. Commonly, network 108 topographies are divided into segments (subnets) for management and other purposes. These different segments typically assign different network (transport) addresses to the various Mobile End Systems 104 within the given segment.

It is common to use a Dynamic Host Configuration Protocol (DHCP) to automatically configure network devices that are newly activated on such a subnet. For example, a DHCP server on the sub-net typically provides its

clients with (among other things) a valid network address to "lease". DHCP clients may not have permanently assigned, "hard coded" network addresses. Instead, at boot time, the DHCP client requests a network address from the DHCP server. The DHCP server has a pool of network addresses that are available for assignment. When a DHCP client requests an network address, the DHCP server assigns, or leases, an available address from that pool to the client. The assigned network address is then owned" by the client for a specified period ("lease duration"). When the lease expires, the network address is returned to the pool and becomes available for reassignment to another client. In addition to automatically assigning network addresses, DHCP also provides netmasks and other configuration information to clients running DHCP client software. More information concerning the standard DHCP protocol can be found in RFC2131.

Thus, when a Mobile End System 104 using DHCP roams from one subnet to another, it will appear with a new network address. In accordance with one aspect of the present invention, Mobile End Systems 104 and Mobility Management Server 102 take advantage of the automatic configuration functionality of DHCP, and coordinate together to ensure that the Mobility Management Server recognizes the Mobile End System's "new" network address and associates it with the previously-established connection the Mobility Management Server is proxying on its behalf.

One example embodiment uses standard DHCP Discover/Offer client-server broadcast messaging sequences as an echo request-response, along with other standard methodologies in order to determine if a Mobile End System 104 has roamed to a new subnet or is out of range. In accordance with the standard DHCP protocol, a Mobile End System 104 requiring a network address will periodically broadcast client identifier and

hardware address as part of a DHCP Discover message. The DHCP server will broadcast its Offer response (this message is broadcast rather than transmitted specifically to the requesting Mobile End System because the Mobile End System doesn't yet have a network address to send to). Thus, any Mobile End System 104 on the particular subnet will pick up any DHCP Offer server response to any other Mobile End System broadcast on the same subnet.

This example embodiment provides DHCP listeners to monitor the DHCP broadcast messages and thereby ascertain whether a particular Mobile End System 104 has roamed from one subnet to another and is being offered the ability to acquire a new network address by DHCP. Figure 16 shows example DHCP listener data structures. For example, a Mobile End System listener data structure 902 may comprise:

- a linked list of server data structures,
- an integer transaction ID number (xid),
- a counter ("ping"), and
- a timeout value.

A server data structure 904 may comprise a linked list of data blocks each defining a different DHCP server, each data block comprising:

- a pointer to next server,
- a server ID (network address of a DHCP server),
- an address (giaddr) of a BOOTP relay agent recently associated with this DHCP server,
- a "ping" value (socket -> ping), and
- a flag.

These data structures are continually updated based on DHCP broadcast traffic appearing on network 108. The following example functions can be used to maintain these data structures:

- roamCreate() [initialize variables]
- roamDeinitialize() [delete all listeners]
- roamStartIndications() [call a supplied callback routine when a Mobile End System has roamed or changed interfaces, to give a registrant roaming indications]
- roamStopIndications() [remove the appropriate callback from the list, to stop giving a registrant roaming indications]
- Interface Change [callback notification from operating system indicating an interface has changed its network address]
- Listener Signal [per-interface callback from a Listener indicating a roaming or out-of-range or back-in-range condition].

Additionally, a refresh process may be used to update Listeners after interface changes.

In the preferred embodiment, all Mobile End Systems 104 transmit the same Client Identifier and Hardware Address in DHCP Discover requests. This allows the listener data structures and associated processes to distinguish Mobile End System-originated Discover requests from Discover requests initiated by other network devices. Likewise, the DHCP server will broadcast its response, so any Mobile End System 104 and/or the Mobility Management Server 102 will be able to pick up the DHCP server Offer response to any other Mobile End System. Since multiple DHCP servers can respond to a single DHCP Discover message, the listener data structures shown in Figure 16 store each server response in a separate data block, tied to the main handle via linked list.

Upon receiving a Discover request having the predetermined Client Hardware Address and Client Identifier, the preferred embodiment recognizes this request as coming from a Mobile End System 104. If the message also has a BOOTP relay address set to zero, this indicates that the message originated on the same subnet as the listener. Listeners may ignore all DHCP Offers unless they have a transaction ID (xid) matching that of a Discover message recently sent by a Mobile End System 104. The listener can determine that a Mobile End System 104 has roamed if any response comes from a known server with a new BOOTP relay agent ID and/or offered network address masked with an offered subnet mask. Listeners add new servers to the Figure 16 data structures only after receiving a positive response from an old server. If a listener receives responses from new server(s) but none from an old server, this may indicate roaming (this can be a configurable option). If the listener fails to receive responses from new or old servers, the listener is out of range (this determination can be used to signal an upper layer such as an application to halt or reduce sending of data to avoid buffer overflow).

If the listener never receives a response from any server, there is no point of reference and thus impossible to determine whether roaming has occurred. This condition can be handled by signaling an error after a timeout and allowing the caller to retry the process. The preferred embodiment determines that a Mobile End System 104 has roamed if any response has come from a known server with a new BOOTP relay agent ID (or a new offered network address when masked with offered subnet mask). If the listener data structures see responses from new servers but none from an old server, it is possible that roaming has occurred, but there must be a delay before signaling, in order to wait for any potential responses from the old servers. If there are no responses from new or old servers, then the

Mobile End System 104 is probably out of range and Mobility Management Server 102 waits for it to come back into range.

Figure 17 is a flowchart of example steps of a Listener process of the preferred embodiment. Referring to Figure 17, a DHCP listener process is created by allocating appropriate memory for the handle, opening NAL sockets for the DHCP client and server UDP ports, and setting receive callbacks for both. A timer is then set (block 802) and then the process enters the "Wait" state to wait for a roaming related event (block 804). Three external inputs can trigger an event:

- a DHCP server packet is received;
- a DHCP client packet sent by another Mobile End System is received
- a timer timeout occurs.

If a DHCP server packet has been received, the packet is examined to determine whether its client identifier matches the predetermined client ID (decision block 806). If it does not, it is discarded. However, if the packet does contain the predetermined ID, a test is performed to determine whether the packet is a DHCP Offer packet (decision block 808). Offer packets are rejected unless they contain a transaction ID matching a recently sent DHCP Discover sequence.

If the packet transaction ID matches (block 810), then a test is made as to whether the server sending the DHCP offer packet is known (i.e., the server ID is in the listener data structure shown in Figure 16) (block 812). If the server ID is not on the list ("no" exit to decision block 812), it is added to the list and marked as "new" (or "first" if it is the first server on the list) (block 822). If the server is already on the list ("Y" exit to decision block 812), a further test is performed to determine whether the packet BOOTP relay address ("GIADDR") matches the server address

("GIADDR") (decision block 814). If there is no match, then the Offer packet must be originating from a different subnet, and it is determined that a "hard roam" has occurred (block 816). The caller application is signaled that there has been a roam. If, on the other hand, decision block 814 determines there is a match in BOOTP relay addresses, then no roam has occurred, the listener process stamps the server receive time, resets "new" flags for all other servers on the list, and stores the current ping number with the server (block 818, 820). The process then returns to "wait" period.

If the event is a received client packet, the listener process determines whether the packet has the predetermined client ID, is a DHCP Discover packet and has a BOOTP relay address (GIADDR) of 0 (blocks 824, 826, 828). These steps determine whether the received packet is DHCP Discover message sent by another Mobile End System 104 on the same sub-net as the listener. If so, the listener process then sets the transaction ID to the peer's transaction ID (block 830) for use in comparing with later-received DHCP Offer packets, calls a ping check (block 834) and resets the timer (block 836).

In response to a timer timeout, the process calls a "ping check" (block 838). "Pings" in the preferred embodiment are DHCP Discover packets with a random new xid. Example steps for this ping check 838 are shown in Figure 17A. The purpose of the ping check routine is to determine if a "soft roam" condition has occurred (i.e., a Mobile End System has temporarily lost and then regained contact with a sub-net, but has not roamed to a different sub-net). The process determines whether there is a sub-net roam condition, an out-of-range condition, or a "no server" condition. In other words:

- Has a Mobile End System roamed from one sub-net to another?
- Is a Mobile End System out of range?

- Is a DHCP server absent?

These conditions are determined by comparing Mobile End System prior "ping" response with the current "ping" response (decision blocks 846, 850). For example, if the current ping number minus the old server's last ping response is greater than the sub-net server pings and there is at least one server marked "new," there has been a sub-net roam to a different server. The result of this logic is to either signal a subset roam, and out of range condition or a no server condition (or none of these) to the calling process.

Figure 18 shows a flowchart of example steps performed by a Mobile End System 104 roaming control center. To enable roaming at the Mobile End System 104, the list of known addresses is initialized to zero (block 850) and an operating system interface change notification is enabled (block 852). The process then calls the operating system to get a list of current addresses that use DHCP (block 854). All known addresses no longer in the current list have their corresponding listeners closed (block 856). Similarly, the process opens listeners on all current but not known interfaces (block 858). The process then signals "roam" to registrants (block 860).

When the listener process of Figure 17 signals (block 862), the process determines whether the signal indicates a "roam", "out of range" or "back in range" condition (decision block 864, 870, 874). A roam signal ("yes" exit to decision block 864) causes the process to close corresponding listener 866 and call the operating system to release and renew DHCP lease to a network address (block 868). If the listener signals "out of range" (decision block 870), the process signals this condition to registrants (block 872). If the signal is a "back in range" (decision block 874), then this condition is signaled to all registrants (block 876). Upon receiving a

disabled roam command (block 878), the process closes all listeners (block 880) and disables the operating system interface change notification (block 882).

Example Interface Assisted Roaming Listener

A further, interface-based listener feature enables roaming across network points of attachment on the same network or across different network media. This interface-based listener feature operates without requiring the beaconing techniques described above, while permitting the system to fall back on beaconing if the underlying interface(s) is unable to support the appropriate signaling.

In this further embodiment, an interface-based listener integrates information from network interface adapters (e.g., via a low level interface roaming driver) with information available from network stacks to determine whether a mobile node has moved to a new Network Point of Attachment. Figures 19A & 19B show an example listener algorithm that may be used to efficiently determine the migration path of the mobile node. This process is shown using a single network interface connected to a single network medium, but can be used by itself or in conjunction with other roaming algorithms to traverse across many diverse network media and interfaces (e.g., to create a self-healing infrastructure using redundant paths).

Referring to Figure 19A, at system initialization time or when the network adapter driver loads (Figure 19A, block 2000), low-level interface roaming drivers register with the roaming control center module of Figure 18 (block 2010). Such registration (which is made via the function crRegisterCardHandler() in the example embodiment) provides entry points for:

- open,
- close,
- get status, and
- a Boolean set to TRUE if the driver can notify the registrant of changes in status, and FALSE if the roaming control center module should use timer-based (or other) polling to check status.

The example embodiment function crRegisterCardHandler() also provides a interface description string or token that can be used by the roaming control center module for preliminary match-ups to the correct roaming driver. A default roaming driver may also be installed for interfaces that use an O/S generic mechanism for signaling/querying media connectivity as well as changes to network point of attachments.

In the example embodiment, when an interface's state becomes enabled (i.e. access to the network is now possible) (block 2020), the roaming control center tries to enable Interface Assisted Roaming (IAR) according to the following steps (please note however, that the steps may be interchanged or either might be omitted based on the design of the operating system (O/S) and/or the hosting device being used in a particular application):

1. If a generic handler is installed, a call to the generic crOpenInstance() handler is made. The generic handler queries the low-level adapter driver to see if it can generically support signaling the status of media connectivity as well as any changes to the network point of attachment (block 2030). If the interface driver is unable to support this functionality generically ("no" exit to block 2030), an error status is returned to the caller to indicate that it should use an alternative mechanism for acquiring signaling information.

2. If the generic handler returns an error ("no" exit to block 2030), a search is made with the token of the activated interface through the currently registered roaming drivers (block 2040). If the interface matches one of the tokens that was registered during crRegisterCardHandler() phase (block 2050), the roaming control center calls the specific crOpenInstance() for that instance of the adapter. This function attempts to open the low level driver, poll once for status (media connectivity, and the network point of attachment ID), and set the periodic polling timer (if applicable). If the low-level driver does not support the requests for some reason, an error is returned indicating that the roaming control center should use an alternate mechanism for acquiring signaling information.

3. If either of the previous steps is unable to achieve the required functionality, an error is returned to the roaming control center to signal that it should not use the IAR functionality and fall back to other roaming algorithms, such as the beaconing listener shown in Figure 17 & 17A, Mobile IP, or in some cases the currently attached network itself deals with roaming ("no" exit to block 2050, block 2999). Otherwise Interface Assisted Roaming is enabled (block 2060) and the roaming control center follows the algorithm outlined below.

Initially, the interface-assisted listener records current media connectivity status and network point of attachment identification information in a local data store (block 2060). Assuming the interface assisted subsystem is successful in providing roaming feedback, the subsystem waits for a status event (block 2100). The event can comprise, for example:

- a callback from the low level roaming driver,
- a timed poll interval (blocks 2070, 2090), or

- a hint from network level activity (i.e. trouble transmitting/receiving) (block 2080).

If the status of the interface signifies either a change in medium connectivity has occurred, or a change in network point of attachment ("yes" exit to block 2110 or 2120 of Figure 19B), any clients of the roaming control center are notified of the state change using the following rules:

1. If the status signifies a change from being connected to the underlying network medium to being detached ("yes" exit to block 2120) and there are no other paths to the peer, the listener concludes that the mobile end system has lost its connection, and the roaming control center signals its clients with a status of ROAM_SIGNAL_OUT_OF_CONTACT (block 2140).

2. If the status signifies that the interface has been reconnected to the medium, and the network point of attachment has not changed ("no" exit to block 2150 after "no" exit to block 2120) and a ROAM_SIGNAL_OUT_OF_CONTACT was previously signaled, this indicates that the mobile end system had previously lost but has now reestablished contact with a particular network point of attachment. In this case, the roaming control center will revalidate any network address it may have registered or acquired for proper access (block 2170), and signals ROAM_SIGNAL_ROAM_SAME_SUBNET (block 2180) to alert the roaming control center clients that a reattachment has occurred and that they should take whatever steps necessary to quickly reestablish transport level communications. For example, during the disruption in service it is possible that some data may have been lost -- and the clients may need to act to recover such lost data.

3. If the status signifies that the interface is attached to the medium but the network point of attachment has changed ("yes" exit to block 2150),

the roaming control center will signal its clients that a roaming condition has occurred. To more efficiently support handoffs between network point of attachments, the roaming control center in this example employs the use of a learning algorithm along with a local data-store. The data-store is normally populated dynamically (i.e. learning), but it can be seeded with static information (i.e., already learned information) to improve performance. The data-store itself maintains a list of network points of attachment identifiers, along with information such as network and media access address, network mask, etc. This "network topology map" assists the roaming control center in deciding the correct signal to generate to its clients.

Determination of the correct signal is done in the following manner in the example embodiment:

a) A search is made through the network topology map data-store to determine if the interface has already visited this particular network point of attachment (block 2190). If a match is found ("yes" exit to block 2200), a further check is made to see if the network point of attachment is on the same network segment as the one that the interface was previously associated with. If the network segment is the same, the roaming control center generates a ROAM_SIGNAL_ROAM_SAME_SUBNET. This alerts the roaming control center clients that a handoff occurred and it should take whatever steps necessary to quickly reestablish transport level communications as during the handoff it is possible that some data may have been lost.

b) If during the search a match is found, but the new network point of attachment is not on the same network segment, then the listener concludes that the mobile end system has roamed to a different subnetwork. In this case, the roaming control center:

- acquires an address that is usable on the new network segment (block 2220). This may entail registering the current address to be valid on the new segment, (re)acquiring an address from a local server, having one statically defined, or using heuristics to determine that an address that was previously assigned is still valid. In the latter case, the roaming control center may determine that the interface is changing between a given set of network point of attachments and may not immediately relinquish or de-register the network address for performance reasons. In this example, there is a difference between acquiring an address on the network (e.g., via DHCP) or registering the address on the local network (e.g., via a foreign agent in Mobile IP). The roaming entity either (re)acquires (e.g., possibly establishing/updating a lease with the DHCP server) or registers the current address with a foreign agent (Mobile IP).
- Generates a ROAM_SIGNAL_ROAM signal to its clients (block 2230) indicating roaming to a different subnet.

c) If the search yields no match ("no" exit to block 2200), a new record is created in the local data-store populated with the network point of attachment's identifier, media access address, network mask and other ancillary information (block 2210). The roaming control center then executes blocks 2220 and 2230 to acquire and register a network address, and to generate a "roam" signal.

Since the interface-assisted roaming technique described above gives access to the underlying interface information, it is possible to employ an additional set of policy parameters (defined by the user and/or the system) that can enable automatic efficient selection of alternate valid network paths. If there is more than one network available at a time, the subsystem

can choose the path(s) with the least cost associated with it (i.e., a wide area network connection versus a local area connection). This can be done by a number of metrics such as, for example, bandwidth, cost (per byte), and/or quality of service. Such "least cost routing" techniques can provide advantages in terms of network connection quality, efficiency, and reduction in frame loss. For example, it is possible to provide a "make before break" handoff scheme based on other heuristics available (media connectivity, signal strength, retransmission rate, etc.), thus allowing continuous packet flow with minimal loss to and from the roaming node. See policy management discussion below.

Figure 20 shows an example interface assisted roaming topology node data structure. Figure 20 shows this data structure implemented as a linked list, but it could alternatively be represented as an array where the next and previous fields are omitted. In a wireless network infrastructure, the "NPOA" may, for example, be the MAC address of the access point or base station that the mobile node is associated with. In other networks, it may be the unique identifier of an intervening network interconnect (e.g., gateway, IWF, etc.). The data structure may be seeded with static information or dynamically learned. Other information may also be associated with each node (e.g., MTU size, latency, cost, availablilty, etc.)

EXAMPLE FURTHER EMBODIMENT TO HANDLE CERTAIN RACE CONDITIONS

Through further experimentation evidence has shown that some network adapters may erroneously signal that they are (re)connected to the medium before they are totally registered on the network segment. In some instances during roaming events the storage area of where the network identifier is kept may not yet been updated, and thus it is possible for the

system to incorrectly believe that these adapters have roamed back onto the same subnet. Eventually, when the device finishes registering, the storage area is updated with the new network identifier, causing yet another ROAM signal to be generated. This scenario would correctly work if both pieces of information were gated together and only signaled once when the interface was finished registering with the network. However when polling it is difficult to determine when the network ID is valid if the "in contact with network" signal is generated previously.

In essences the roaming node may in fact be in media connectivity state since it can communicate at the media access level with the network, but in fact one cannot yet send any application data across the link since the registration process has not completed. Therefore, it is desirable to compensate for this condition. One way to provide such compensation is to determine peer connectivity by sending link confirmation frames, or what is more commonly known as an echo request/response packets. These echo or ping frames are generated by one peer (most likely the roaming node), to determine if two-way peer-to-peer connectivity is achievable. If the requesting peer receives a response frame to its request, it can be concluded that a duplex path has been achieved. At this point, the NPOA information can be regarded as valid until the next disconnect situation is realized. Other information, such as the reception of any frame from the peer on the interface in question, also allows the roaming node to assume the registration process has concluded and two-way communications is achievable.

Another race condition between the network interface and the underlying protocol stack situation has arisen that can sometimes cause a problem. It is possible for a device to have roamed to a new network segment and been signaled correctly from the interface below, but the

transport stack itself has not made the necessary adjustments to its routing table(s) for application data to flow. To compensate for this condition, an additional signal ROAM_SIGNAL_ROUTE_CHANGE, was added and is generated whenever the underlying transport's routing table changes. When this signal is indicated, the roaming subsystem clients take whatever action is necessary to determine if connectivity to the peer systems is achievable. This may entail the roaming client to enumerate through the underlying transport's routing table to determine if the routing modification has affected the communications path to the peer. Other more intrusive algorithms, such as the ones described above, can also be used to confirm that a two-way communication path exists between the peers.

Example Roaming Across Disjoint Networks

A further aspect of an example non-limiting preferred embodiment of our invention provides an algorithm and arrangement for accessing the MMS (Mobility Management Server) in what we call "disjoint networking" mode. The new algorithm allows for dynamic/static discovery of alternate network addresses that can be used to establish/continue communications with an MMS -- even in a disjoint network topology in which one network may have no knowledge of network addresses for another network.

In general, the algorithm allows for a list of alternate addresses that the MMS is available at to be forwarded to an MES (Mobile End System) during the course of a conversation. Thus, the MMS uses a connection over one network to send the MES one or more MMS network addresses or other MMS identities corresponding to other networks. As one example, this list can sent during circuit creation. It is also possible for the list to change midstream. In this case, the list can be updated at any time during the connection.

If/when the MES roams to another network, it uses the list of MMS "alias" addresses/identifications to contact the MMS from the new network point of attachment. This allows the MES to re-establish contact with the MMS over the new network connection even though the primary and ancillary networks may not share any address or other information.

Figure 21 shows a simplified flowchart of this new technique. Suppose that the MMS 102 is connected to two different disjoint networks or network segments N1 and N2. Suppose that the MES 104 is initially coupled to the MES 102 via network N1. Once a connection has been established between the MES 104 and the MMS 102 over network N1, the MMS 102 can send the MES 104 a list L of network addresses or other identifiers that the MMS is called on one or more other networks (e.g., network N2). The MES 104 receives and stores this list L. Then, when the MES 104 roams to another network (N2), it can access this stored list L and use it to efficiently re-establish communication with the MMS 102 over the new network (N2).

There are at least several uses for this new algorithm in addition to the ability to more efficiently obtain an alternative network address or other identifier for communicating with the MMS 102 over a disjoint network. One example usage is secure network operation. For example, using the algorithm shown in Figure 21, one can setup a secure network where the MMS 102 is used as a secure firewall/gateway from a multitude of networks (some/all may be wireless) and a corporate backbone, and allow for secure and seamless migration of the mobile node 104 between all disassociated networks. Think, for example, of the MMS 102 as a hub, with one fat pipe connecting to the corporate network and many little spokes connecting many logically discrete networks. Since they are logically discrete, traffic

on one network segment cannot reach another, except through the MMS 102 (which can act as a router in this example).

Normally for a node to roam from network segment to network segment, there must be routing information/paths provided on each network segment (i.e. default route, etc) specifying how to get back to the "main public or initial address" used to contact the MMS 102. Once a connection is established, that address is used for the life of the connection. When a frame is sent from the MES 104, the IP network (layer 3) infrastructure on the client and intermediary nodes (routers) looks at the destination address of the frame and correctly forwards the packet on to its ultimate destination (the MMS 102). This is done by using what is commonly referred to as IP forwarding, or IP routing. With this functionality turned on, frames (broadcasts, etc) from one network segment can leak onto another. By not using IP forwarding, frames sent on one segment are not forwarded onto the other, thus breaking the communications pipe or creating a disjoint network.

The alternate address list shown in Figure 21 has the effect of pushing or distributing some of the routing intelligence out to the MES 104. Each segment therefore can be kept discrete and without knowledge of any other segment attached to the MMS 102. The MES 104 can be authenticated by the MMS 102 so that the MMS only sends a list L to authorized MES units 104. When the MES 104 roams onto another networks segment, it can automatically select the correct address to use to initiate/continue communications with the MMS midstream, thus solving the disjoint network problem, and not require any changes to the routing infrastructure. This provides for a more secure computing environment by only letting validated users to gain access to the network.

For example, by using the MMS 102 in this manner combined with user level security/encryption, we can limit traffic from and to the corporate

backbone to only the frames destined for those nodes on that segment using the roaming techniques described above. Frames can be optionally encrypted to thwart any potential eavesdropping by devices that may be validated by the spoke network infrastructure.

Figure 22 shows an example. In Figure 22, the MMS 102 is attached to four separate and distinct networks (Ia, Ib, Ic, Id) without any interconnects or route information shared. For all intents and purposes, each network I is an island. Now envision an MES 104 being docked to one of the networks (e.g., Ic) using a wired connection on the corporate backbone. For example, suppose that the MES 104 acquires an address on the 192.168.x.x network to communicate with the MMS 102.

Now suppose that for some reason, the MES now needs to migrate or roam to the 10.1.x.x (Ia) network. Since the 10.1.x.x (Ia) network has no knowledge of the 192.168.x.x (Ib) network (i.e. no routes to it), when the MES 104 moves into its domain, the communication pipe is broken even though the MMS is attached to it. Again, the same thing happens when the mobile node 104 attaches to any of the other 10.x networks depicted.

Now using the algorithm shown in Figure 21, the MMS 102 at connection initiation time (or by some other method) shares its interfaces address on each of the various disjoint networks Ia, Ib, Ic, Id with the MES 104 and the MES records these. Once recorded, if the MES 104 roams into any one of the networks and detects that it has roamed onto a new network segment, the MES can now select the appropriate network address to communicate with the MMS for that network segment. If more then one address can be used, the MES 104 can select the appropriate address to use based on a number of metrics such as speed, cost, availability, hops, etc. An MES 104 that has not received a list as in Figure 21 may be effectively

prevented from roaming between the various networks because it has no way to contact the MMS over any network other than its "home" network.

Another application for the Figure 21 technique is in distributed network interfaces. In today's networks, folks have deployed what is known as Network Address Translators (NATs). By use of this conventional technology, one can have many network devices use only one public network address for access to information on the Internet. The technology provides this functionality by funneling all information and queries destined to the Internet through a single/few device(s). The device(s) records the request at the network layer, then remaps the address and port information in the packet to the devices own address/port tuple and sends it onto its destination. Upon reception of a frame from the Internet or other such network, the device(s) does the reverse look and forwards it back to the correct source by replacing its address/port tuple information with that of the initiating device. These mappings may be defined statically also at the NAT.

Suppose someone wants to use the MMS 102 for the LAN/WLAN internally and have it sit behind a NAT. Currently, unless the MMS 102 is the NAT, or by using a different proxy for all communications with the MMS, when someone roams outside of the bounds of the intranet, the MMS is no longer accessible since the address to converse with it is no longer accessible. With the Figure 21 algorithm, one can statically/dynamically define another interface address that is not directly attached to the MMS. Therefore, using the algorithm described above, the MES 104 can now automatically select the appropriate disjoint address to use when attaching to a network that is outside the intranet's domain.

Figure 23 illustrates this scenario. Suppose a node migrates from interface "d" to interface "g". Just supplying the MMS 102 local interfaces

would not allow access. The MES 104 needs a priori knowledge of the distributed interface. It can then select the necessary address to use on interface "g". The NAT 2000 will then do the appropriate translation of network address/port information on each packet to the internal interface "c" address. The reverse operation will happen on frames sent by the MMS 102 to the MES 104.

Example Policy Management and Location Based Services

A further non-limiting embodiment of the invention provides the unique ability to offer additional security, cost savings, and services based on a number of metrics. Since the MMS described above is intimately involved with each application session the MES establishes, either side (i.e., the MMS and/or the MES) can apply policy-based rules to tailor and control the communications between the MES and its ultimate peer. It can further condition or modify applications request based on the locale or proximity of the device and its attachment to the network. For example, the MMS and/or the MES can include a rules engine that applies learned, statically defined, or other rules based on policy to each application session that is established or request that is attempted. The MMS can further distribute some, none or part of such rules and/or processing to the MES to provide further metering or security against rogue attacks of the mobile device. Unlike certain other policy management technology available in a distributed topology, the MMS provides a central place to administer the rules and policy decisions and have them distributed to the remote device at any time during the course of a conversation/connection.

The rules themselves can be configured based on user, user group, device, device group, process, application identity and/or network point of attachment. Once defined (learned), they can be combined to govern and

control a variety of different events, activities, and/or services, including for example:

- denying, allowing or conditioning ingress access to the remote device;
- denying, allowing or conditioning access to specific network resources based on identity,
- denying, allowing or conditioning access to available or allowable bandwidth,
- denying, allowing or conditioning access to other network resources and/or
- modifying, conditioning or changing content or information.

Such decisions can be based on any of various different factors including for example:

- proximity, location, altitude and/or other characteristics of the mobile device,
- time of day,
- application or process identity, address, etc;
- application behavior (e.g., bandwidth requirements);
- current network conditions; and/or
- other static or dynamic factors

Furthermore by employing the distributed architecture, the MMS can also apply or share the same decision set. Having the MMS perform the policy management processing and/or decision making may be desirable in instances where the mobile device has limited processing power to execute the engine or bandwidth limitations are applicable, or for security purposes.

Figure 24 shows an example table of the some metrics (rules) that might be used to control a sample MES. This table may be populated either

statically or dynamically, and maybe updated anytime before, during, or after the connection/conversation. For example, a person could use a rules editor (e.g., a wizard) other mechanism to define entries in the table. In other example arrangements, the metrics could be automatically defined by the system based on learning, or could be dynamically changed based on changing conditions. The rules also have a priority assigned to them whether implied by the location in the table or specifically designated by an assignment. This priority allows the engine to correctly determine the expected behavior. Additional user interface functions allow the system administrator and or user of the device to interrogate the rules engine and test out the functionality of a given rule set.

The Figure 24 example table shows a number of example metrics on which policy management decisions may be based, including:

- MES communications capability (transmit only, receive only, or transmit and receive);
- Whether the MES request is proxied;
- MES source port;
- MES source address;
- MES destination port;
- MES destination address;
- MES protocol;
- Amount of bandwidth available;
- Process name(s), identities or other characteristics;
- Network name(s), identities or other characteristics;
- Location (e.g., GPS coordinates or other location information);
- Network point of attachment;
- User identity name, identity or other characteristic;

- Other metrics.

It will be appreciated that the invention should not be limited by the scope of the metrics entries in the example table as it is not meant to be an exhaustive list. The entries can be specific as in this example or use a generic mechanism (e.g., wildcards) to describe the desired behavior of the mobile node with regards to network access and entitlements.

The Figure 24 example table further includes a "deny request" entry that indicates the result of a policy management decision to be made based on the metrics. As one example, the particular example entries in the Figure 24 table specify that all connections to destination ports 20 and 21 should be denied or throttled back if the available bandwidth is reduced to less than 100,000 bytes per second. Furthermore, in the particular example shown, rules (rows) 3 and 4 allow only network traffic to flow to and from the MMS (all other network traffic that is not proxied is implicitly discarded).

In one example, before each RPC request or frame is processed, the rules engine is consulted to determine if the status of the operation. Base on the outcome of this process, the request may be allowed, denied or delayed. Figure 25 is an example flowchart of steps that may be performed by the MMS and/or the MES to make policy management decisions.

Furthermore by combining the roaming technology outlined previously with other location or navigational information that mat be available, the MMS detects when a mobile end system has moved from one point of attachment to another. By combining this information in conjunction with the ability of the mobile end system to detect a change in environment of network topology, or locale enable the invention provides additional levels of location based monitoring and services.

To fully realize the potential of this information, enhancements to both the Internet Mobility Protocol and RPC engine are outlined. Several

new RPC protocol and configuration enhancements will be added to provide this functionality. These are listed below.

Example Location Change RPC

When the mobile end system has determined that it has moved to a new point of attachment using interface assisted roaming or some other method such as detecting changes from a global positioning system, it will send a formatted "Location Change RPC Request" message to its peer, in this case the mobility management server. The "Location Change RPC" formats one or more of the point of attachment identification information into a type, length, value format. The type identifies the kind of identification information, types supported will include but will not be limited to 48 bit IEEE MAC Addresses, IPV4 Addresses, IPV6 Addresses, longitude, latitude, altitude, and attachment names in ASCII. The length indicates the length in bytes of the identification data, and the data contains the actual point of attachment identification. The mobility management server upon receipt of the "Location Change RPC Request" will build a "Location Change Alert" that contains the point of attachment identification and other pertinent information such as the mobile end system identification, the user name, and PID. It then will forward the alert to the alert subsystem on the server. The alert will be formatted with the same type, length, data format utilized within the "Location Change RPC Request". The alert subsystem will then forward the location change alert with this information to all applications that have registered for the alert. Applications that have registered for the alert may include monitoring applications such as the current active status monitor, a long-term activity log, the policy management engine, and other third party applications and network management tools. One such third party application may combine

this location information with Web based maps to provide detailed information about a mobile end system's or MMS location. In addition to such applications, other actions can be associated with location change alerts. This includes sending an email, printing a message, launching a program and/or change in policy.

The Location Change RPC will contain a field in its header that indicates if it was triggered due to location change, distance change, or rate change.

In some instances, the MES may not know it has roamed. Depending on the medium and the network adapter it is attached to, the MMS may be the only entity that notices that the MES has migrated to a new point of attachment. Consider the case of a mobile router. The addresses behind the router stay the same, only the routers address changes. In this case, the MMS knows the new care of address of the MES. Therefore, for complete motion detection it needs to be a combination of the both the MES and MMS to detect motion. In the present embodiment, the MMS detects motion of the clients at the IMP layer when the source address changes and a new IMP message is received. When this occurs, the MMS locally generates a Location Change Alert. It also sends a message back to the MES that its point of attachment has changed.

<u>Example Topology RPC</u>

The "Topology RPC Request" is sent from the mobility management server to mobile end systems. Upon receipt of this RPC the mobile end system will read the topology information stored in its local data store and build a Topology RPC Response. The Topology RPC response will be formatted with a Total Length Field followed by consecutive type, length, data point of attachment identification followed by type, length, value data

indicating the subnet and network information. This information may be used on the server to build a complete topological map of the mobile network being served by the server.

<u>Example Location Information UI</u>

The user interface on the server will provide a method for mapping and displaying location information. This location information will be available for each active mobile end system and the long-term activity log will maintain a history of all active and previously active mobile end system location changes. The user interface will permit the system administrator to configure the point of attachment information in human readable form. For example, if the point of attachment information is provided in the form of a 48-bit IEEE MAC address this MAC address will be displayed along with the information provided through the user interface on the server. If the point of attachment represented an access point in front of the "HallMark Cards" store it might be configured to present the following information "HallMark, Street Address, City, State, Zip". When displayed to the user, information "HallMark, Street Address, City, State, Zip" is presented.

<u>Example Location RPC Timer</u>

A configurable timer is provided on the mobile end system to limit the rate at which Location Change RPCs may be sent from the mobile end system to the mobility management server. If the timer interval is larger than the rate at which the point of attachment changes are occurring, the mobile end system will wait until the timer interval expires before generating another Location Change RPC.

Example Distance Change Notification

A distance metric will be provided for triggering the generation of Location Change RPCs. This setting configures the system to send an update when the user moves three dimensionally every n feet from, kilometer, or other appropriate unit of measure from the last point of origin. By default this setting is disabled. Enabling this setting causes a Change Notification when the distance interval in the configuration is exceeded.

Example Rate Threshold Notification

A rate change metric will be provided for triggering the generation of Location Change RPCs. This parameter is configured in distance per second such as miles per hour. It will specify an upper and lower bounds and a time interval that the attained rate must be sustained (i.e. 0 MPH for 10 minutes or 70 MPH for 1 minute). When this speed is reached a Location Change Notification will be generated.

EXAMPLES

The present invention finds application in a variety of real-world situations. For example:

Intermittently Connected Portable Computer

Many businesses have employees who occasionally telecommute or work from home. Such employees often use laptop computers to get their work done. While at work, the employees typically connect their laptop computers to a local area network such as an Ethernet through use of a docking port or other connector. The LAN connection provides access to network services (e.g., printers, network drives) and network applications (e.g., database access, email services).

Now suppose an employee working on a project needs to go home for the evening and wants to resume working from home. The employee can "suspend" the operating system and applications running on the laptop computer, pack up the laptop computer, and bring the laptop computer home.

Once home, the employee can "resume" the operating system and applications running on the laptop computer, and reconnect to the office LAN via a dialup connection and/or over the Internet. The Mobility Management Server (which continued to proxy the laptop computer vis-a-vis the network and its applications during the time the laptop computer was temporarily suspended) can re-authenticate the laptop computer and resume communicating with the laptop computer.

From the perspective of the employee now working from home, all of the network drive mappings, print services, email sessions, database queries, and other network services and applications, are exactly where the employee left them at the office. Furthermore, because the Mobility Management Service continued to proxy the laptop computer's sessions, none of those network applications terminated the laptop computer's sessions during the time the employee was traveling from the office to home. The invention thus provides efficient persistence of session across the same or multiple network mediums that is very powerful and useful in this and other contexts.

Mobile Inventory and Warehouse Application

Imagine a large warehouse or retail chain. Within this campus, inventory workers use vehicle mounted (i.e., trucks and forklifts) personal laptop computers and handheld data collection units and terminals to perform inventory management of goods. Warehouse and retail workers are

often inexperienced computer users that do not understand network sub-nets and require management supervision. The present invention allows the creation of a turnkey system that hides the complexity of the mobile network from the warehouse users. The users can move in and out of range of access points, suspend and resume their Mobile End Systems 104, and change locations without concern for host sessions, network addresses, or transport connections. In addition, the management software on the Mobility Management Server 102 provides management personnel with metrics such as number of transactions, which may be used to gauge worker productivity. Management can also use the network sub-net and access points to determine worker's last known physical location.

Mobile Medical Application

Imagine a large hospital using radio LAN technology for network communications between several buildings. Each building is on a unique sub-net. The present invention enables nurses and doctors to move from room to room with handheld personal computers or terminals -- reading and writing patient information in hospital databases. Access to the most recent articles on medication and medical procedures is readily available through the local database and the World Wide Web. While in the hospital, pagers (one and two way) are no longer required since the present invention allows continuous connection to the Mobile End System 104. Messages can be sent directly to medical personnel via the Mobile End System 104. As in the case with warehouse workers, medical personnel are not required to understand the mobile network they are using. In addition, the Mobile End System 104 allows medical personnel to disable radio transmission in area where radio emissions are deemed undesirable (e.g., where they might

interfere with other medical equipment) -- and easily resume and reconnect where they left off.

Trucking and Freight

Freight companies can use the present invention to track inventory. While docked at a warehouse, the Mobile End System 104 may use LAN technology to update warehouse inventories. While away from local services, the Mobile End System 104 can use Wide Area WAN services such as CDPD and ARDIS to maintain real time status and location of inventory. The Mobile End System 104 automatically switches between network infrastructures -- hiding the complexity of network topology from vehicle personnel.

Mobile Enterprise

Corporate employees may use the system in accordance with the present invention for access to E-mail, web content and messaging services while within an enterprise campus that has invested in an infrastructure such as 802.11. The cost of ownership is reduced since pager service and other mobile device services are no longer required. The purchase of mobile infrastructure is a one time capital expense as opposed to the costly "pay-per-use" model offered by many existing mobile device services.

IP Multiplication

If an organization has a LAN that needs to be connected to the Internet, the administrator of the LAN has two choices: get enough globally assigned addresses for all computers on the LAN, or get just a few globally assigned addresses and use the Mobility Management Server 102 in accordance with the present invention as an address multiplier. Getting a large number of IP addresses tends to be either expensive or impossible. A

small company using an Internet Service Provider (ISP) for access to the Internet can only use the IP addresses the ISP assigns -- and the number of IP addresses limits the number of computers that can be on the Internet at the same time. An ISP also charges per connection, so the more computers that need to be on the Internet, the more expensive this solution becomes.

Using the Mobility Management Server 102 in accordance with the present invention as an address multiplier could solve many of these problems. The enterprise could put the Mobility Management Server 102 on hardware that is connected to the Internet via an ISP. Mobile End Systems 104 could then easily connect. Because all connection to the Internet would go through the Mobility Management Server 102, only one address from the ISP is required. Thus, using the present invention as an address multiplier allows the enterprise to get just a few (in many cases one) addresses and accounts from the ISP, and allows the entire LAN to have simultaneous connections to the Internet (assuming enough bandwidth is provided).

While the invention has been described in connection with what is presently considered to be the most practical and preferred embodiment, it is to be understood that the invention is not to be limited to the disclosed embodiment, but on the contrary, is intended to cover various modifications and equivalent arrangements included within the scope of the appended claims.

WHAT IS CLAIMED IS:

1. A mobile computing network including at least one mobile computing device coupled to the network via a network point of attachment, characterized by a policy-management arrangement that applies policy management rules based on various metrics including mobile computing device location.

2. A network as in claim 1 further characterized in that processing of the attributes of the rules can be distributed and applied at either the mobile computing device and or the mobility management server or both.

3. A network as in any of the preceding claims further characterized in that prioritization of the rules is either implied by position in the entry in such table or explicitly noted by an ordinal ensuring the expected behavior.

4. A network as in any of the preceding claims further characterized in that datastore for the rule attributes is locally or centrally administered via central management services

5. A network as in any of the preceding claims further characterized in that behavior of a particular application(s) is modified based on a number of metrics including cost of service, network point of attachment, trust relationship, etc.

6. A network as in any of the preceding claims further characterized in that the effect of the behavior modification is to allow, deny or delay a request based on attributes of the rules.

7. A network as in any of the preceding claims further characterized in that even if the application is already started, a rule or set of rules is invoked to modify the application(s) processes.

8. A network as in any of the preceding claims further characterized in that point of presence information (location) is further used to govern application behavior or provide relevant information to the mobile computing device.

9. A network as in any of the preceding claims further characterized in that rate of motion along with distance measurements is used to alter behavior of applications or the communication path.

10. A network as in any of the preceding claims further characterized in that topological information is extracted and displayed as result of the location information.

11. In a mobile computing network including at least one mobile computing device coupled to the network via a network point of attachment, an improvement comprising an interface-assisted roaming listener that detects, based at least in part on identification of the network point of attachment, whether said mobile computing device has roamed to a different network segment.

12. A network as in claim 11 wherein said mobile computing device includes a network interface adapter, and said listener obtains said network point of attachment identification from said network interface adapter.

13. A network as in claim 11 wherein said listener maintains a network topology map storing information that correlates said network point of attachment identification with further information concerning a network connection.

14. A network as in claim 11 wherein said listener detects when communications with said network is interrupted or reestablished.

15. A network as in claim 14 wherein said listener generates a roam signal in response to detection of (a) network communications interruption and re-establishment, and (b) change of said network point of attachment identification.

16. An interface-based listener for use in a mobile computing device, said interface-based listener integrating information from at least one network interface adapter with information available from at least one network stack to determine whether said mobile computing device has moved to a new network point of attachment.

17. The interface-based listener of claim 16 including a network topology map providing network connection information including network points of attachment information.

18. The listener of claim 17 wherein said listener dynamically constructs said network topology map based on learned information.

19. The interface-based listener of claim 16 including a status checker that checks status based on occurrence of an event.

20. The interface-based listener of claim 16 wherein said event comprises any of a timer timeout, a low level roaming driver callback, and a network level activity hint.

21. The interface-based listener of claim 16 including a connection information searcher that queries an interface as to whether the mobile computing system has already visited the current network point of attachment.

22. The interface-based listener of claim 16 including a connection arrangement that registers or reacquires a current address to be valid on a new network segment.

23. The interface-based listener of claim 16 including a roam signal generator that generates a roam signal in response to detection, based at least in part on information provided by an interface, that the mobile computing device has roamed to a different network segment.

24. The interface-based listener or claim 23 further including a heuristic analyzer that determines whether a previously assigned address is still valid.

25. A method of determining whether a mobile node has moved to a new network point of attachment, comprising:

(a) receiving network point of attachment identification information from a network interface;

(b) using said network point of attachment identification information to determine whether said mobile node has moved to a new network point of attachment; and

(c) generating signaling in response to said step (b).

26. A method as in claim 25 further including maintaining a network topology map, and using said map to perform step (c).

27. A method as in claim 25 wherein said step (c) includes generating a roam signal.

28. A method as in claim 25 wherein said step (b) includes obtaining said network point of attachment information from a network adapter.

29. A method as in claim 25 further including falling back to an alternative roaming detection mechanism if a network interface is not available that supports generic signaling.

30. A method as in claim 25 further including selecting, at least in part in response to said network point of attachment information, between alternate network connection paths.

31. A method for facilitating communication with a mobile system over disjoint networks comprising:

establishing communications between a node and said mobile system over a first network;

sending to the mobile system over the first network, data identifying the node on at least a second network disjoint from the first network; and

using the data to establish communication between the mobile system and the node over the second network.

32. The method of claim 31 further including authenticating the mobile system for authorization to communicate with the node over the second network before sending the data to the node over the first network.

33 The method of claim 21 wherein the sending step comprises sending distributed interface data to the mobile system over the first network.

34. A network as in claim 12 wherein said mobile computing device network interface adapter is physically attached to said network.

35. A network as in claim 12 wherein said mobile computing device communicates wirelessly with the network point of attachment.

36. A method for maintaining communication between a mobile computing system and a network node as the mobile computing system roams between over plural disjoint networks comprising:

establishing communications between the mobile system and a node via a first network segment;

sending the mobile computing system, via the first network segment, information for use in re-establishing communications with said node via plural further network segments each of which are disjoint from the first network segment; and

using said information to re-establish communications between the mobile computing system and the node via any of said plural further, disjoint network segments.

37. The process of claim 36 wherein said information comprises distributed interface data.

38. A process for providing least cost routing in a network having plural disjoint segments, comprising:

(a) establishing communications between the network and a temporarily-attached mobile computing device;

(b) using a roaming mechanism to allow the temporarily-attached mobile computing device to roam between said plural disjoint segments; and

(c) enforcing at least one policy parameter to enable efficient automatic selection of alternate valid network paths for re-establishing communication between the network and the mobile computing device in response to mobile computing device roaming.

39. The process of claim 38 wherein the policy parameter comprises an element selected from the following group: bandwidth, cost per data unit and quality of service.

40. In a mobile computing network including at least one peer computing system and at least one mobile computing device coupled to the network via a physical link, an improvement comprising a server coupled to the network, said server proxying communications between the mobile computing device and the peer computing system so as to maintain a continuous virtual data stream connection between the mobile computing device and the peer computing system during times when the physical link to the mobile computing device is temporarily interrupted.

41. A network as in claim 40 wherein said mobile computing device has a point-of-presence address on said network, said peer computing

system communicates with said server using a virtual address, and said server maps said virtual address to said point-of-presence address.

42. A network as in claim 41 wherein said server detects when said mobile computing device has changed its point-of-presence address, and re-maps said virtual address to said changed point-of-presence address.

43. A network as in claim 40 wherein said server queues and responds to requests from said peer computing system on behalf of said mobile computing device during times when said mobile computing device is temporarily unreachable or roaming.

44. A network as in claim 40 wherein said server communicates with said mobile computing device using a conventional transport protocol.

45. A network as in claim 44 wherein said server communicates with said mobile computing device using remote procedure calls.

46. A network as in claim 44 wherein said server communicates with said mobile computing device using an Internet Mobility Protocol.

47. A network as in claim 46 wherein said Internet Mobility Protocol provides for automatic removal of datagrams based on user-configurable timeouts.

48. A network as in claim 46 wherein said Internet Mobility Protocol provides for automatic removal of datagrams based on user-configurable retries.

49. A network as in claim 40 wherein said server performs per-user policy management of consumption of network resources by said mobile computing device.

50. A network as in claim 40 wherein said server provides user-configurable session priorities for said sessions of said mobile computing device.

51. A network as in claim 40 wherein said mobile network includes plural sub-networks, and said mobile computing device uses Dynamic Host Configuration Protocol along with other methodologies to allow said mobile computing device to roam from one of said plural sub-networks to another of said plural sub-networks.

52. A network as in claim 40 wherein said server comprises a Mobility Management Server.

53. A network as in claim 40 further including at least one mobile interconnect coupling said mobile computing devices to said server.

54. A method of maintaining a persistent connection with at least one mobile computing device in a mobile computing environment, said method including:

managing at least one session between said mobile computing device and at least one further computing device, and

maintaining the session when the mobile computing device becomes unreachable, suspends or changes network address.

55. A method as in claim 54 further including providing at least one user configurable session priority for said session.

56. A method as in claim 54 wherein said managing step includes managing consumption of network resources by said mobile computing device.

57. A method as in claim 54 wherein the mobile computing environment includes plural sub-networks, and said maintaining step uses Dynamic Host Configuration Protocol to maintain the session when said mobile computing device roams between said sub-networks.

58. A method as in claim 54 wherein said managing step communicates datagrams with said mobile computing device and automatically removes unreliable ones of said datagrams based on at least one user configurable parameter.

59. A method as in claim 58 wherein said user configurable parameter comprises a timeout.

60. A method as in claim 58 wherein said user configurable parameter comprises a user configurable retry number.

61. A method as in claim 54 further including providing said mobile computing device with a variable point of presence address, and wherein said managing step includes mapping said variable point of presence address to a virtual address, the session being associated with the virtual address.

62. A method as in claim 54 wherein said managing step includes using a Remote Procedure Call protocol to communicate with the mobile computing device.

63. A method as in claim 54 wherein said maintaining step maintains the connection state of said session during interruptions in a physical link connecting said mobile computing device with said mobile computing environment.

64. A method as in claim 54 wherein said managing step includes communicating with said mobile computing device using at least one standard transport protocol.

65. A method as in claim 54 wherein said mobile computing device includes plural application sources, and said managing step includes coalescing data from said plural application sources into a stream, and forwarding said stream.

66. A method as in claim 65 further including demultiplexing said coalesced data from said stream and forwarding said demultiplexed data to plural associated destinations.

67. A method as in claim 65 wherein said stream includes frames, and said coalescing includes dynamically resizing said frames to accommodate a maximum transmission unit of the mobile computing environment.

68. A method as in claim 65 wherein said coalescing includes maintaining semantics of unreliable data, and selecting discarding said unreliable data based on said semantics.

69. A method as in claim 54 wherein said managing step includes providing guaranteed delivery of messages to and/or from said mobile computing device.

70. A method as in claim 54 wherein said managing step includes controlling which network resources are accessible by said mobile computing device.

71. A server for maintaining a persistent connection with at least one mobile computing device in a mobile computing environment including at least one further computing device, said server including:

a session manager that manages at least one session between said mobile computing device and said at least one further computing device, said session manager maintaining the session when the mobile computing device becomes unreachable, suspends or changes network address.

72. A server as in claim 71 wherein said session manager includes a session priority queue that provides at least one user configurable session priority for said session.

73. A server as in claim 71 wherein said session manager includes means for managing consumption of network resources by said mobile computing device.

74. A server as in claim 71 wherein the mobile computing environment includes plural sub-networks, and said session manager uses Dynamic Host Configuration Protocol to maintain the session when said mobile computing device roams between said sub-networks.

75. A server as in claim 71 wherein said session manager communicates datagrams with said mobile computing device and automatically removes unreliable ones of said datagrams based on at least one user configurable parameter.

76. A server as in claim 75 wherein said user configurable parameter comprises a timeout.

77. A server as in claim 75 wherein said user configurable parameter comprises a user configurable retry number.

78. A server as in claim 71 wherein said mobile computing environment provides said mobile computing device with a variable point of presence address, and said session manager maps said variable point of presence address to a virtual address, the session being associated with the virtual address.

79. A server as in claim 71 wherein said session manager uses a Remote Procedure Call protocol to communicate with the mobile computing device.

80. A server as in claim 71 wherein said mobile computing environment includes at least one physical link connecting said mobile computing device with said mobile computing environment, and said session manager maintains the connection state of said session during interruptions in said physical link.

81. A server as in claim 71 wherein session manager communicates with said mobile computing device using at least one standard transport protocol.

82. A server as in claim 71 wherein said mobile computing device includes plural application sources, and said session manager coalesces data associated with said plural application sources into a stream, and forwards said stream.

83. A server as in claim 71 wherein said mobile computing device includes plural application sources, and said session manager demultiplexes coalesced data from said plural application sources and forwards said demultiplexed data to plural associated destinations.

84. A server as in claim 71 wherein session manager communicates with said mobile computing device using frames, and dynamically resizes said frames to accommodate a maximum transmission unit of the mobile computing environment.

85. A server as in claim 71 wherein said session manager maintains semantics of unreliable data, and selectively discards said unreliable data based on said semantics.

86. A server as in claim 71 wherein said session manager provides guaranteed delivery of messages to and/or from said mobile computing device.

87. A server as in claim 71 wherein said session manager places controls on mobile computing environment resources said mobile computing device can access.

88. In a mobile computing environment including a proxy server, a mobile computing device that maintains a persistent virtual connection with at least one further computing device during times when the mobile computing device becomes unreachable, suspends or changes network address, said mobile computing device including:

a transport driver interface, and

a mobile interceptor coupled to said transport driver interface, said mobile interceptor intercepting requests for network services at said transport driver interface, generating Remote Procedure Calls responsive to said requests for network services, and forwarding said Remote Procedure Calls to said proxy server.

89. A mobile computing device as in claim 88 wherein said mobile interceptor includes a session priority queue that provides at least one user configurable session priority.

90. A mobile computing device as in claim 88 wherein said mobile interceptor includes means for managing consumption of network resources by said mobile computing device.

91. A mobile computing device as in claim 88 wherein the mobile computing environment includes plural sub-networks, and the mobile computing device further includes means for using Dynamic Host Configuration Protocol to obtain a point of presence address when said mobile computing device roams between said sub-networks.

92. A mobile computing device as in claim 88 wherein said mobile interceptor communicates datagrams with proxy server and automatically removes unreliable ones of said datagrams based on at least one user configurable parameter.

93. A mobile computing device as in claim 92 wherein said user configurable parameter comprises a timeout.

94. A mobile computing device as in claim 92 wherein said user configurable parameter comprises a user configurable retry number.

95. A mobile computing device as in claim 88 wherein said mobile computing device has an associated a variable point of presence address that said Mobility Management Server maps to a virtual address.

96. A mobile computing device as in claim 88 wherein said mobile interceptor uses a Remote Procedure Call protocol to communicate with the said Mobility Management Server.

97. A mobile computing device as in claim 88 wherein said mobile computing environment includes at least one physical link connecting said mobile computing device with said mobile computing environment, and said mobile interceptor receives updated connection state information of at least one session from said Mobility Management Server after an interruption in said physical link.

98. A mobile computing device as in claim 88 wherein said mobile computing device includes a standard transport protocol handler, and said mobile interceptor communicates with said Mobility Management Server via said standard transport protocol handler.

99. A mobile computing device as in claim 88 wherein said mobile computing device includes plural application sources, and said mobile interceptor coalesces data associated with said plural application sources into a stream, and forwards said stream to said Mobility Management Server.

100. A mobile computing device as in claim 88 wherein said mobile computing device includes plural application destinations, mobile interceptor demultiplexes coalesced data from plural application sources and forwards said demultiplexed data to said plural application destinations.

101. A mobile computing device as in claim 88 wherein mobile interceptor communicates with said proxy server using frames, and

dynamically resizes said frames to accommodate a maximum transmission unit of the mobile computing environment.

102. A mobile computing device as in claim 88 wherein said mobile interceptor maintains semantics of unreliable data, and selectively discards said unreliable data based on said semantics.

103. A mobile computing device as in claim 88 wherein said mobile interceptor provides guaranteed delivery of messages to and/or from said proxy server.

104. A mobile computing device as in claim 88 wherein said mobile interceptor places controls on mobile computing environment resources said mobile computing device can access.

105. A mobile computing environment comprising:
at least one mobile computing device including:
a transport driver interface, and
a mobile interceptor coupled to said transport driver interface, said mobile interceptor intercepting requests for network services at said transport driver interface, generating Remote Procedure Calls responsive to said requests for network services, and forwarding said Remote Procedure Calls to at least one proxy server;
said proxy server including at least one work dispatcher that receives and handles said Remote Procedure Calls forwarded by said mobile interceptor, said proxy server including a proxy queue that proxies a virtual session on behalf of said mobile computing device when the mobile computing device becomes temporarily disconnected from said mobile computing environment.

106. In a mobile computing network including at least one mobile computing device coupled to the network via a network point of attachment, an improvement comprising an interface-assisted roaming listener that detects, based at least in part on identification of the network point of attachment, whether said mobile computing device has roamed to a different network segment.

107. A network as in claim 106 wherein said mobile computing device includes a network interface adapter, and said listener obtains said network point of attachment identification from said network interface adapter.

108. A network as in claim 106 wherein said listener maintains a network topology map storing information that correlates said network point of attachment identification with further information concerning a network connection.

109. A network as in claim 106 wherein said listener detects when communications with said network is interrupted or reestablished.

110. A network as in claim 109 wherein said listener generates a roam signal in response to detection of (a) network communications interruption and re-establishment, and (b) change of said network point of attachment identification.

111. An interface-based listener for use in a mobile computing device, said interface-based listener integrating information from at least one network interface adapter with information available from at least one network stack to determine whether said mobile computing device has moved to a new network point of attachment.

112. The interface-based listener of claim 111 including a network topology map providing network connection information including network points of attachment information.

113. The listener of claim 112 wherein said listener dynamically constructs said network topology map based on learned information.

114. The interface-based listener of claim 111 including a status checker that checks status based on occurrence of an event.

115. The interface-based listener of claim 111 wherein said event comprises any of a timer timeout, a low level roaming driver callback, and a network level activity hint.

116. The interface-based listener of claim 111 including a connection information searcher that queries an interface as to whether the mobile computing system has already visited the current network point of attachment.

117. The interface-based listener of claim 111 including a connection arrangement that registers or reacquires a current address to be valid on a new network segment.

118. The interface-based listener of claim 111 including a roam signal generator that generates a roam signal in response to detection, based at least in part on information provided by an interface, that the mobile computing device has roamed to a different network segment.

119. The interface-based listener or claim 118 further including a heuristic analyzer that determines whether a previously assigned address is still valid.

120. A method of determining whether a mobile node has moved to a new network point of attachment, comprising:

(a) receiving network point of attachment identification information from a network interface;

(b) using said network point of attachment identification information to determine whether said mobile node has moved to a new network point of attachment; and

(c) generating signaling in response to said step (b).

121. A method as in claim 120 further including maintaining a network topology map, and using said map to perform step (c).

122. A method as in claim 120 wherein said step (c) includes generating a roam signal.

123. A method as in claim 120 wherein said step (b) includes obtaining said network point of attachment information from a network adapter.

124. A method as in claim 120 further including falling back to an alternative roaming detection mechanism if a network interface is not available that supports generic signaling.

125. A method as in claim 120 further including selecting, at least in part in response to said network point of attachment information, between alternate network connection paths.

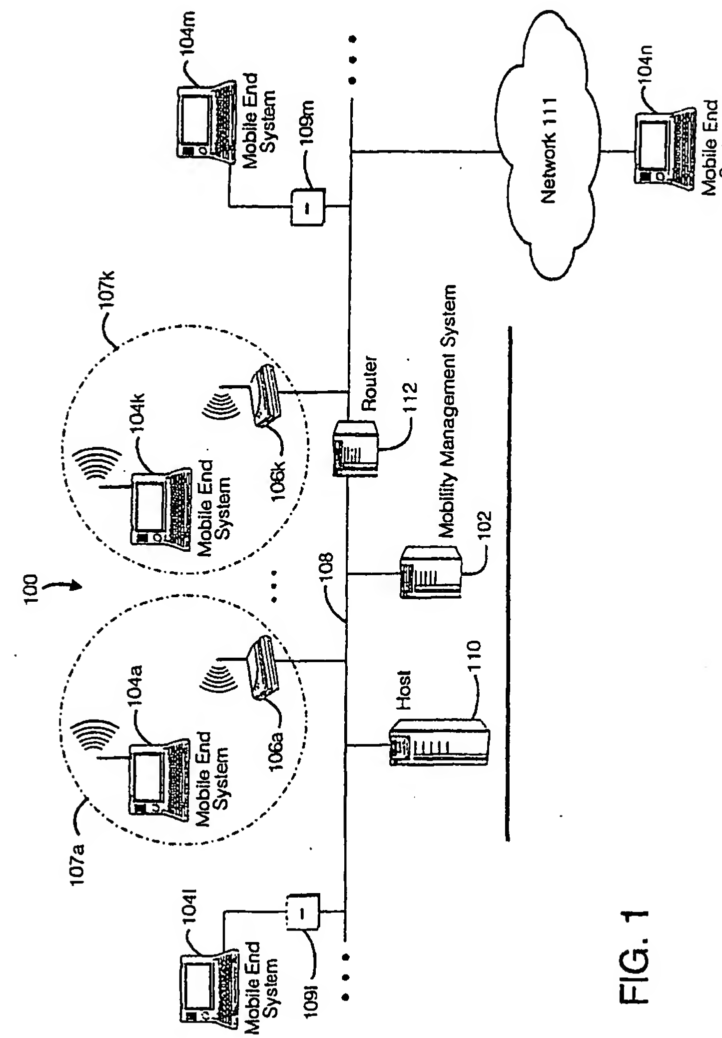


FIG. 1

2/40

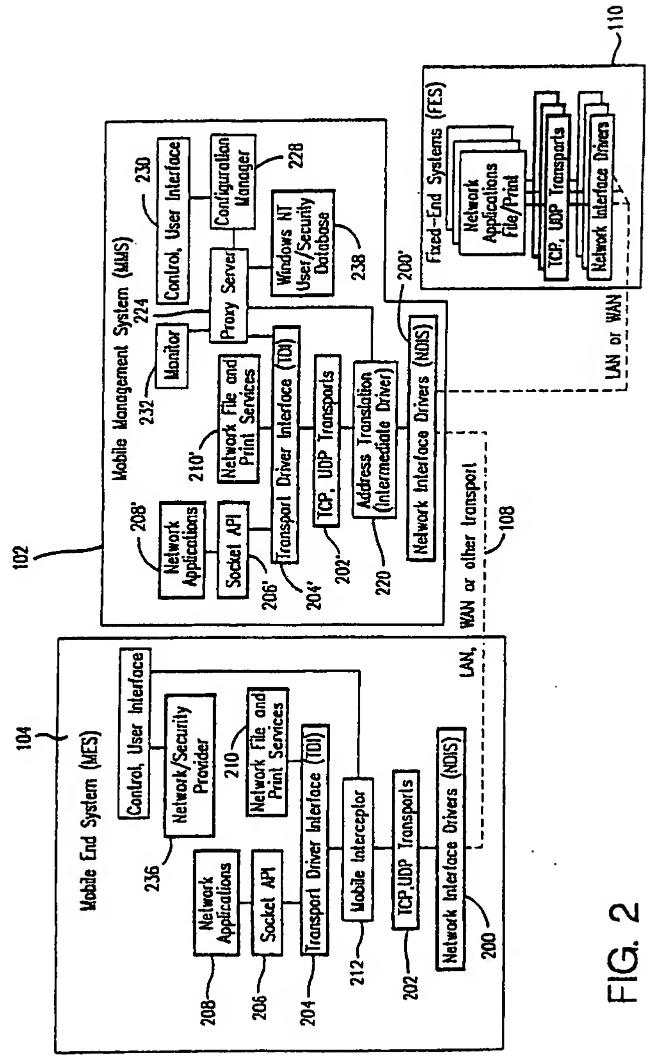


FIG. 2

3/40

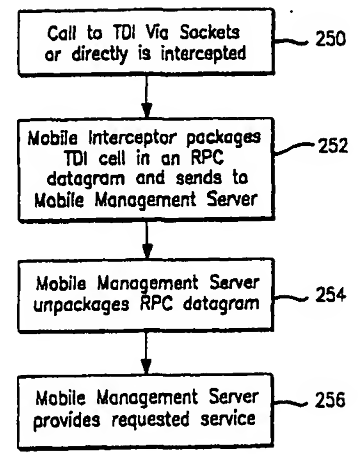


FIG. 2A

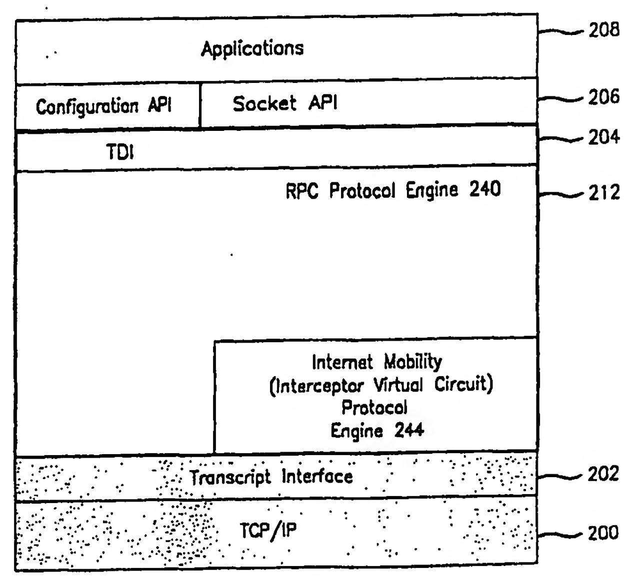


FIG. 3

4/40

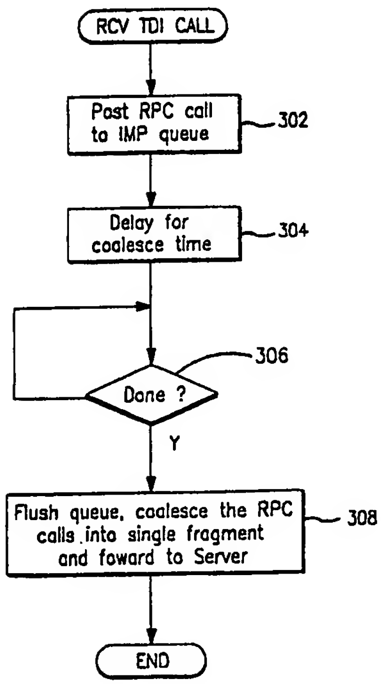


FIG. 3A

5/40

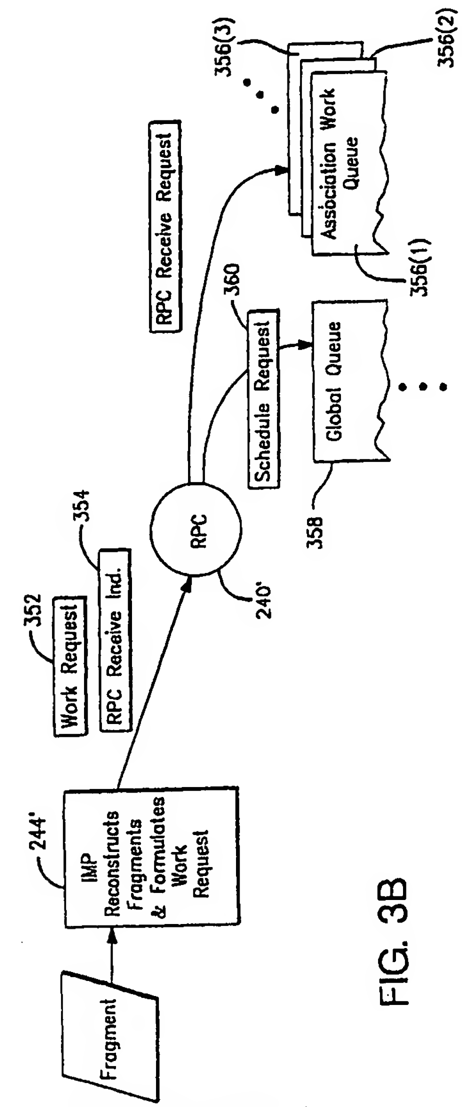


FIG. 3B

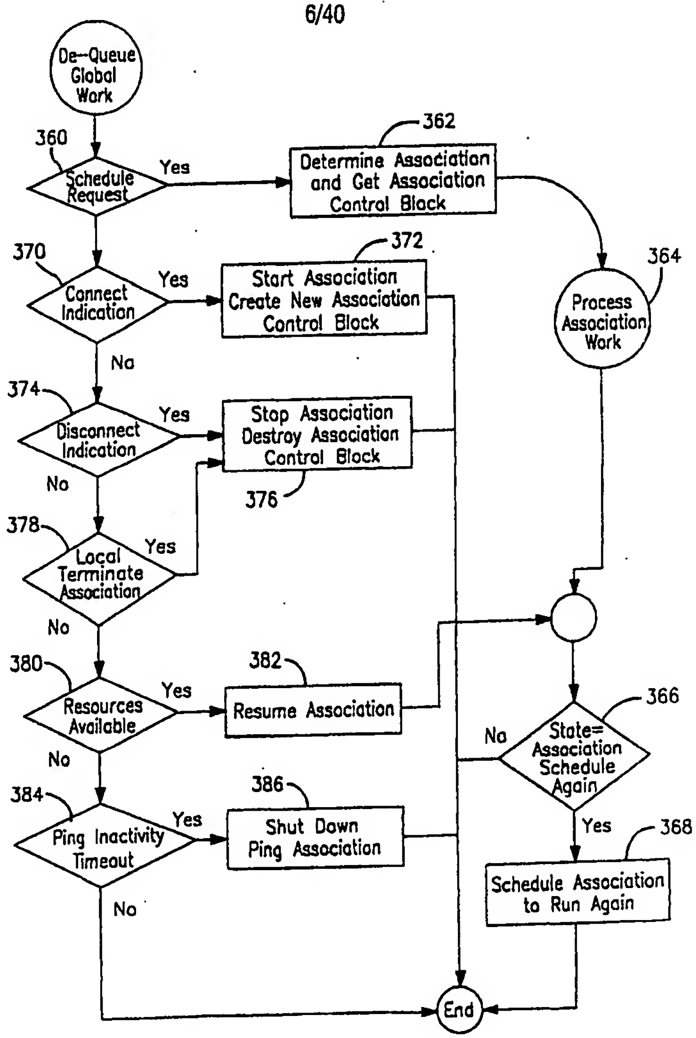


FIG. 4

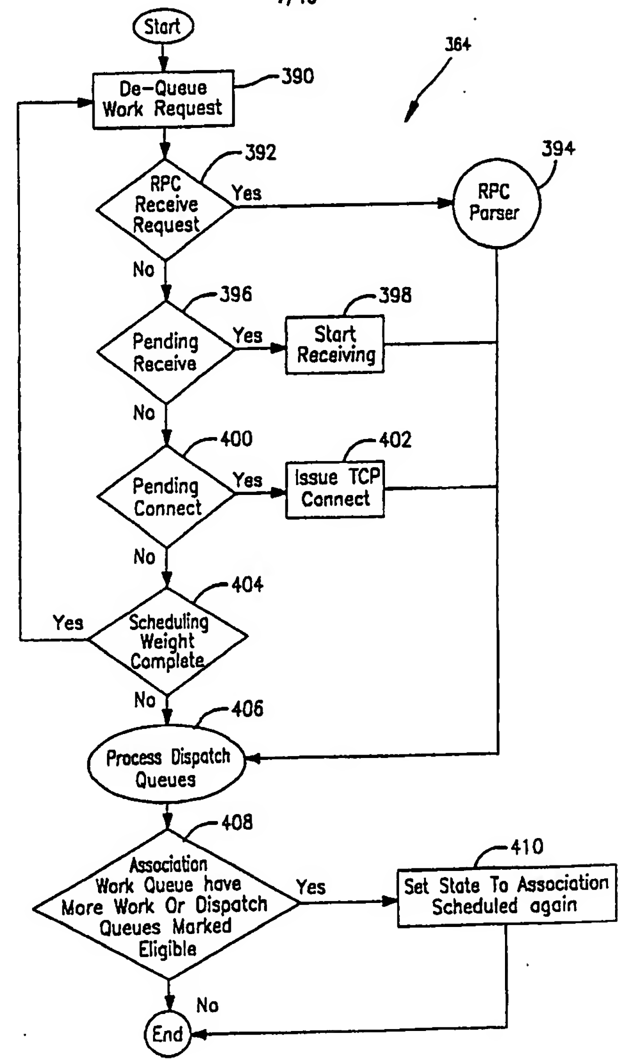


FIG. 5 Process Association Work

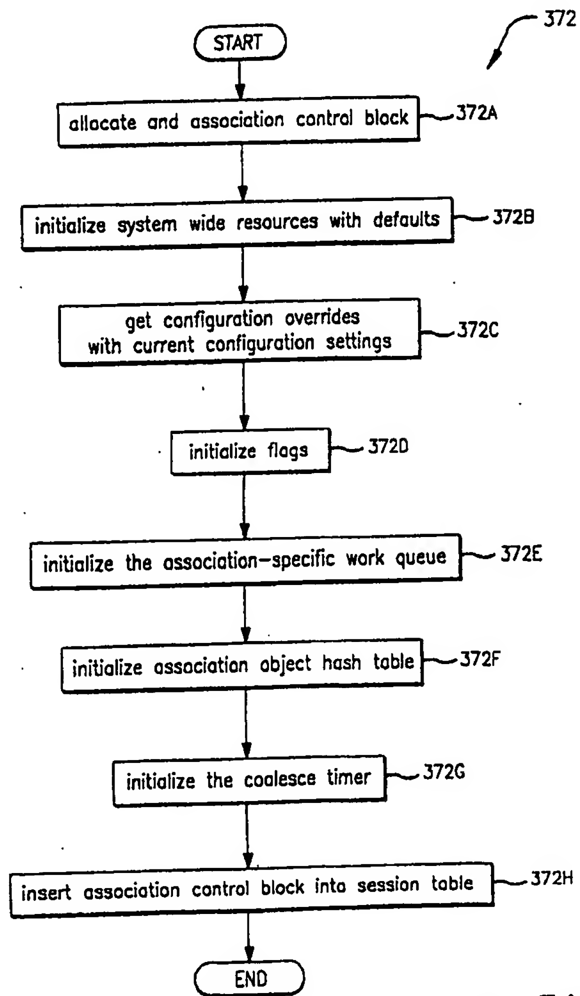

372
START
allocate and association control block
372A
initialize system wide resources with defaults
372B
get configuration overrides with current configuration settings
372C
initialize flags
372D
initialize the association-specific work queue
372E
initialize association object hash table
372F
initialize the coalesce timer
372G
insert association control block into session table
372H
END

FIG. 5A

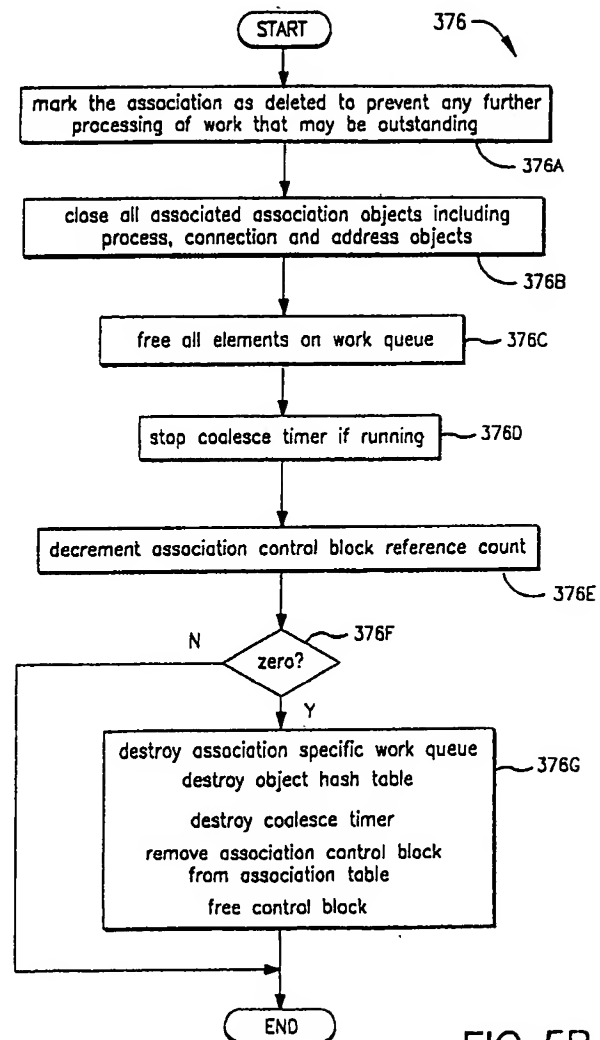

376
START
mark the association as deleted to prevent any further processing of work that may be outstanding
376A
close all associated association objects including process, connection and address objects
376B
free all elements on work queue
376C
stop coalesce timer if running
376D
decrement association control block reference count
376E
376F
zero?
N
Y
destroy association specific work queue
destroy object hash table
destroy coalesce timer
remove association control block from association table
free control block
376G
END

FIG. 5B

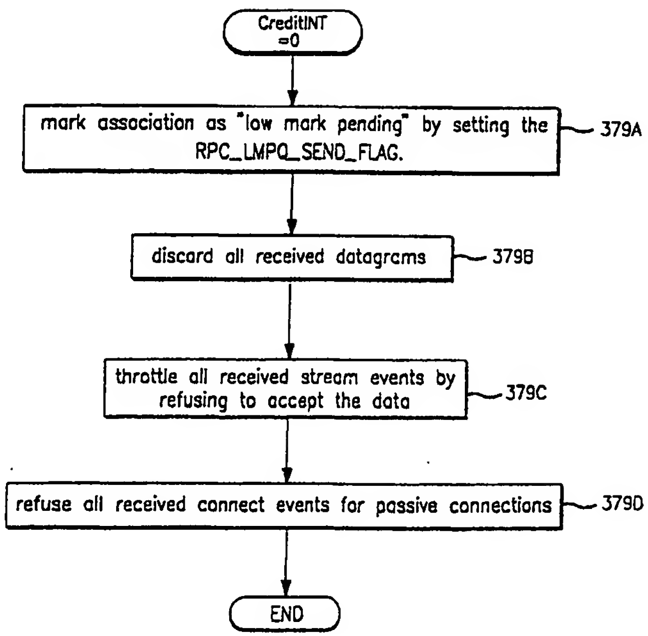

CreditINT =0
mark association as "low mark pending" by setting the RPC_LMPQ_SEND_FLAG.
379A
discard all received datagrams
379B
throttle all received stream events by refusing to accept the data
379C
refuse all received connect events for passive connections
379D
END

FIG. 5C

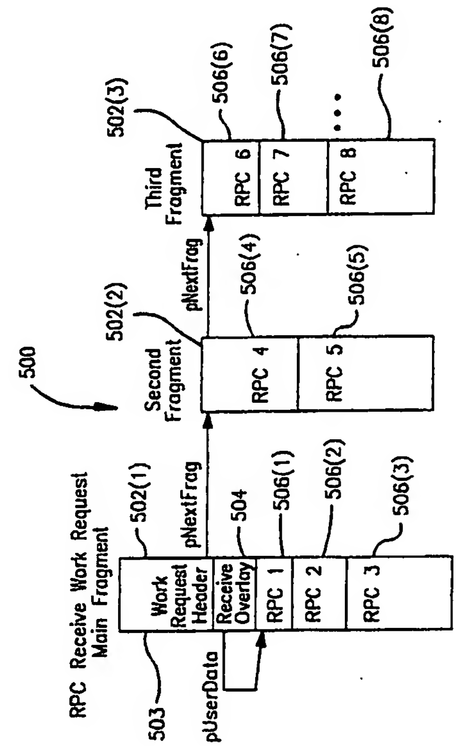

500
RPC Receive Work Request Main Fragment
503
502(1)
Work Request Header
Receive Overlay
504
pUserData
RPC 1
506(1)
RPC 2
506(2)
RPC 3
506(3)
pNextFrag
Second Fragment
502(2)
RPC 4
506(4)
RPC 5
506(5)
pNextFrag
Third Fragment
502(3)
RPC 6
506(6)
RPC 7
506(7)
RPC 8
506(8)

FIG. 6

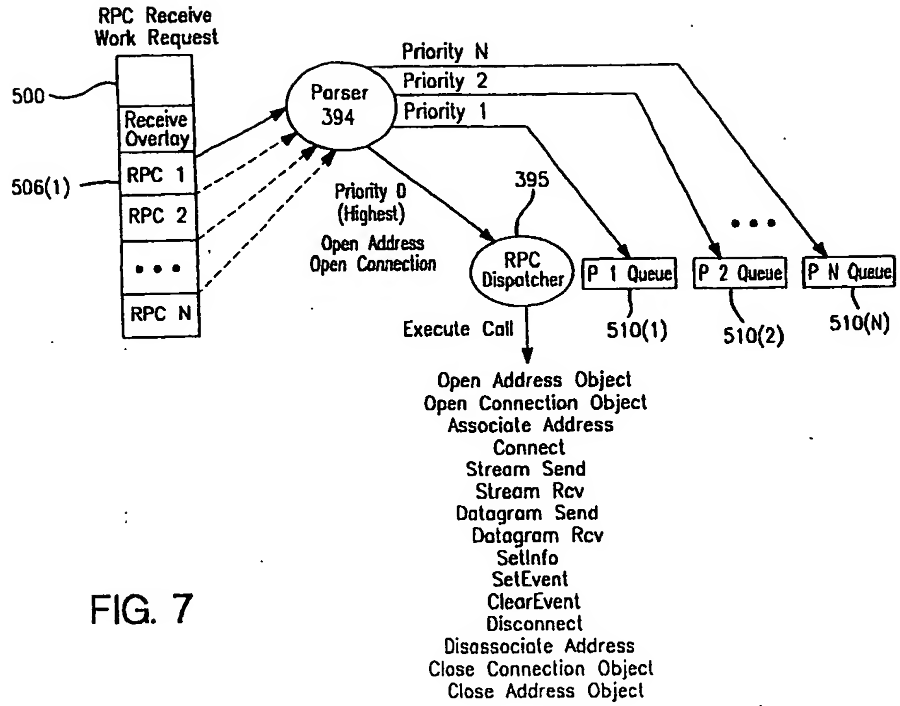
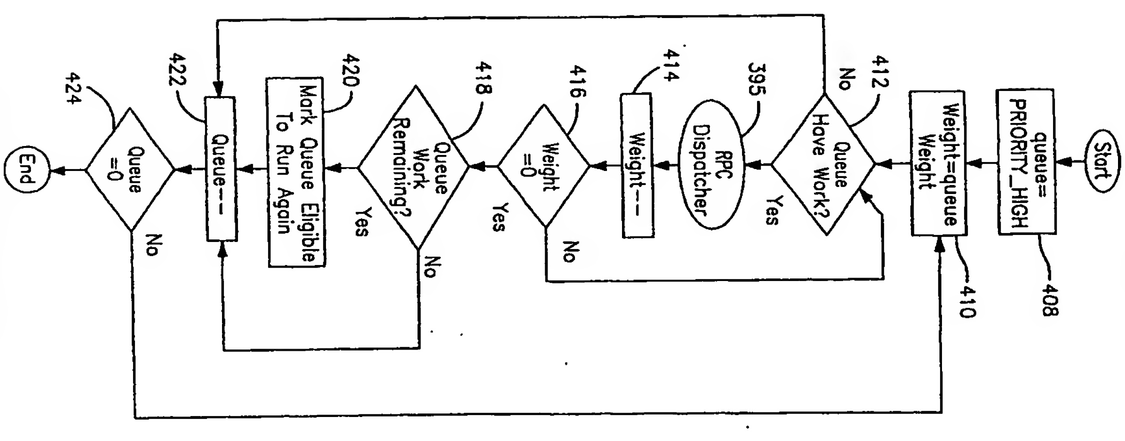


FIG. 7



406

Fig. 8
Process Priority Dispatch Queues

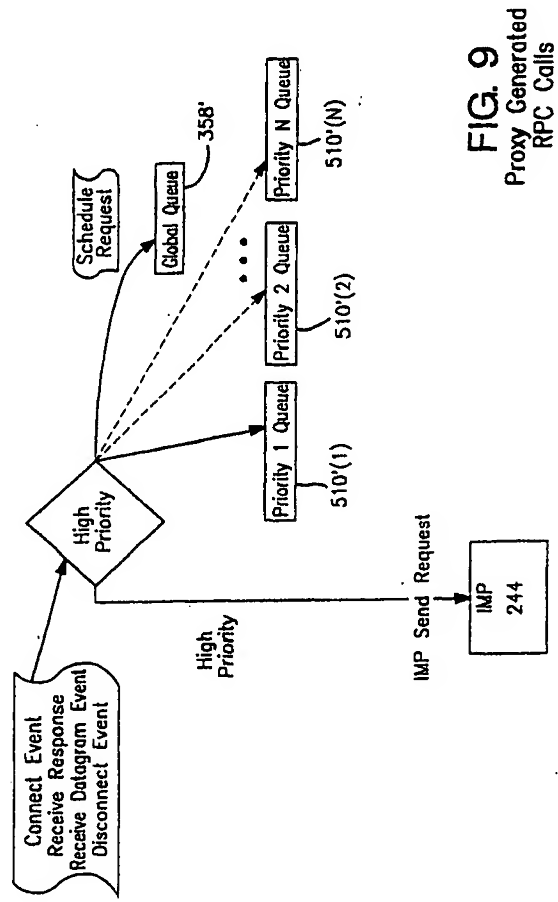
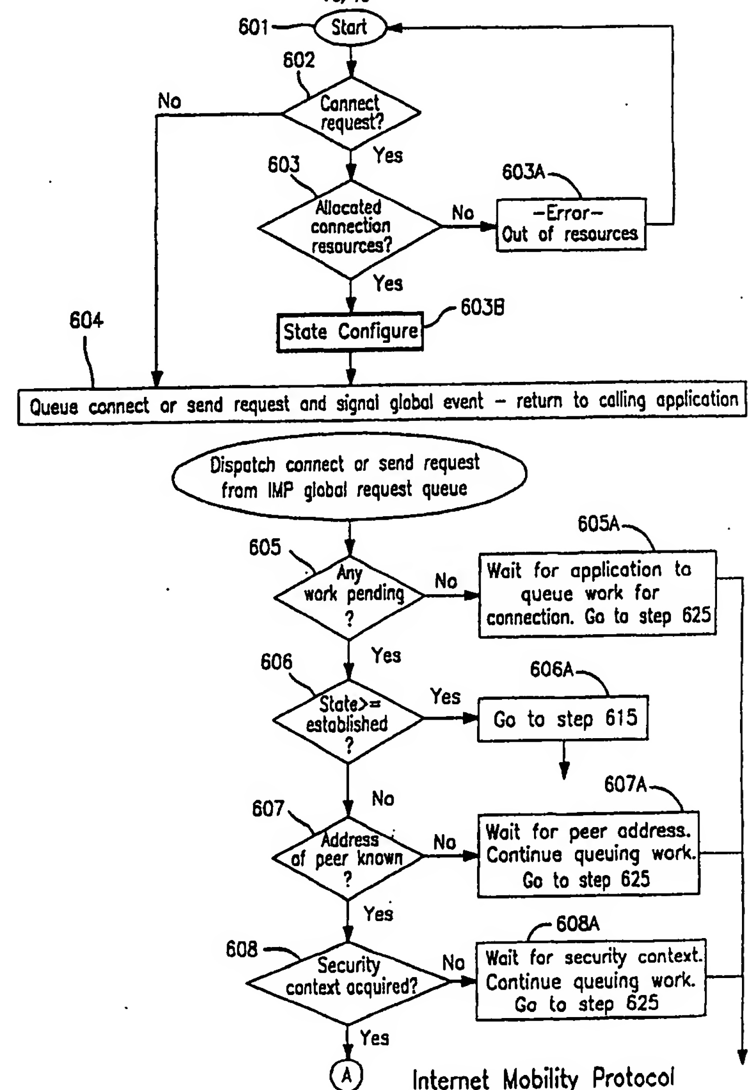


FIG. 9
Proxy Generated
RPC Calls



Internet Mobility Protocol
Connection Decision Tree.
FIG. 10A Connect and Send request logic

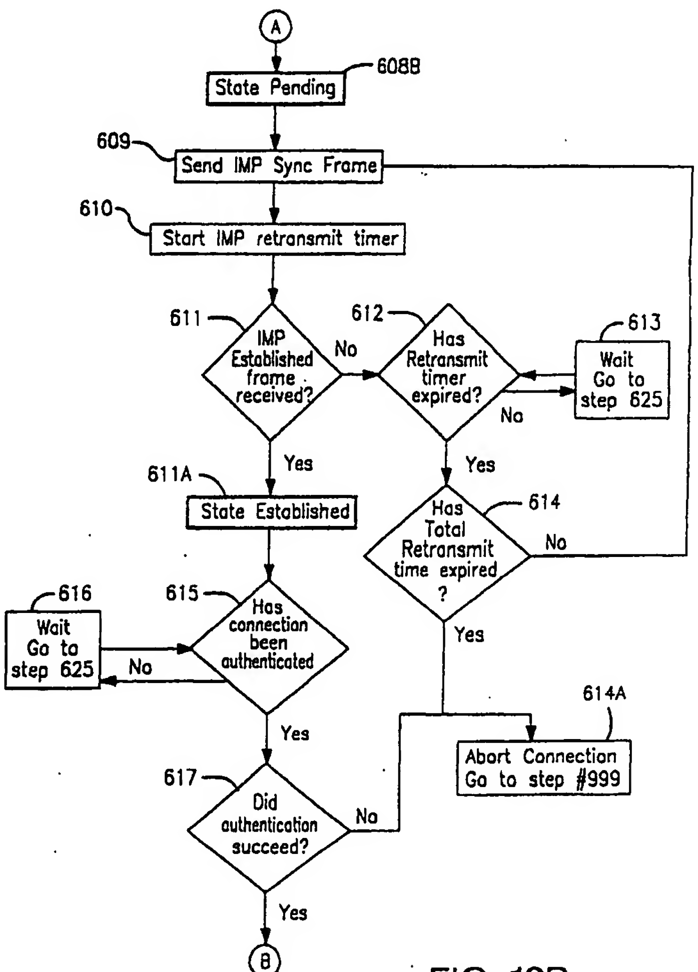


FIG. 10B

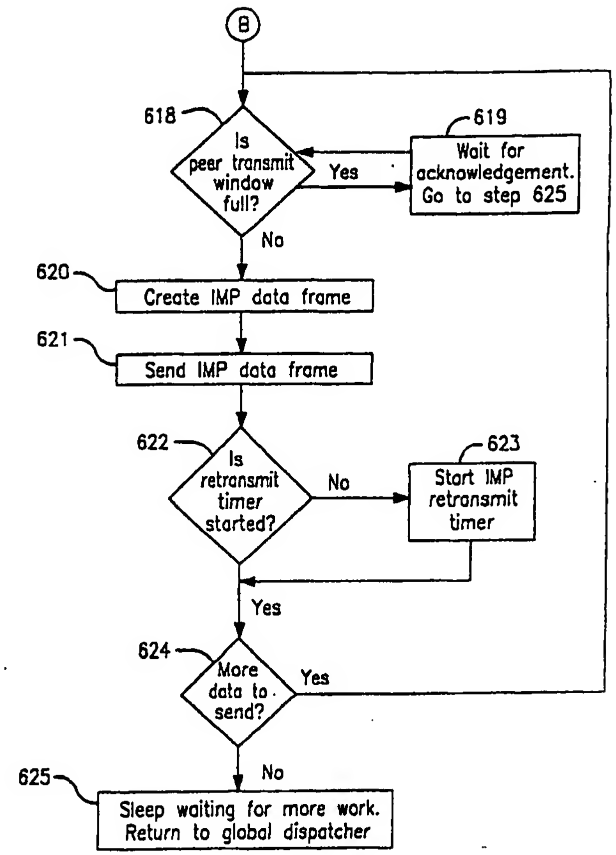


FIG. 10C

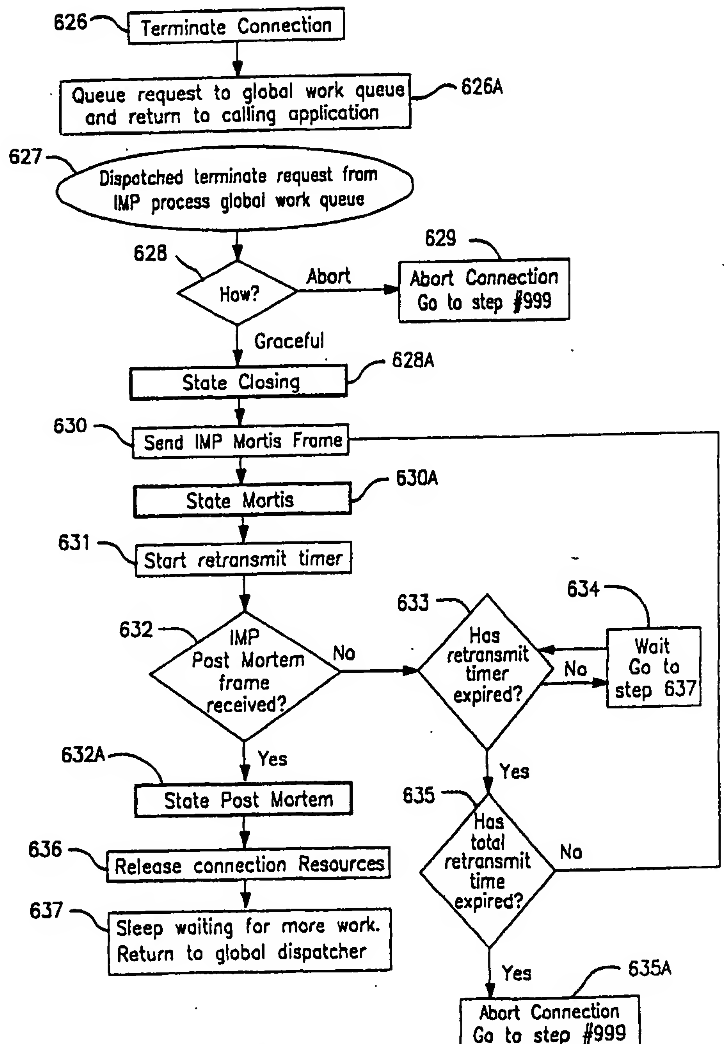


FIG. 11
Terminate Connection request logic

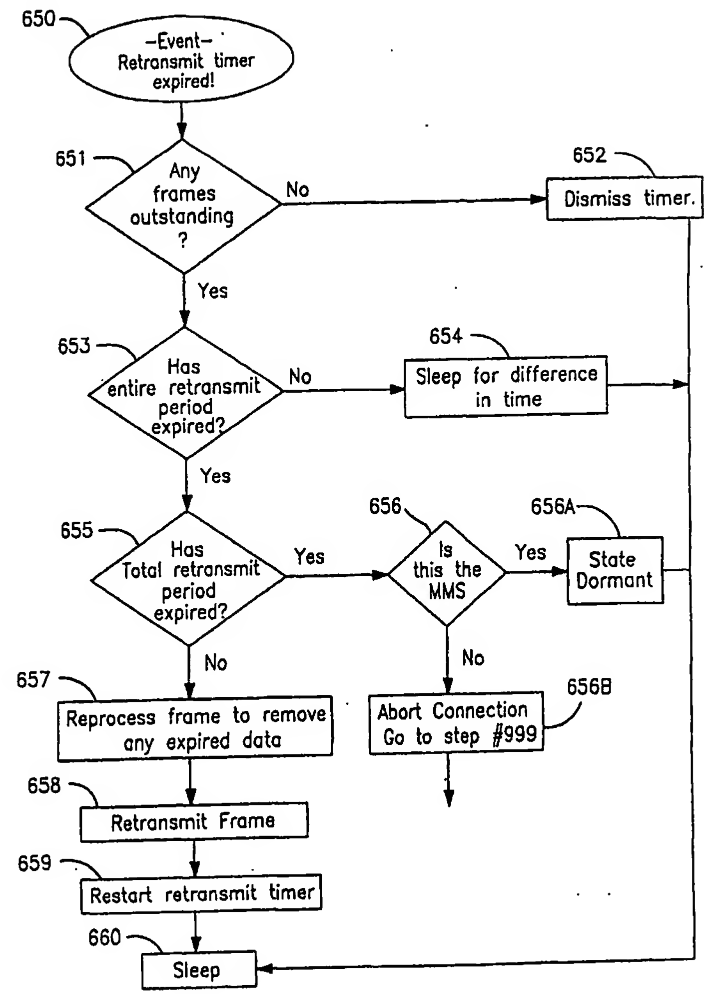


Fig. 12
Retransmit Event Logic

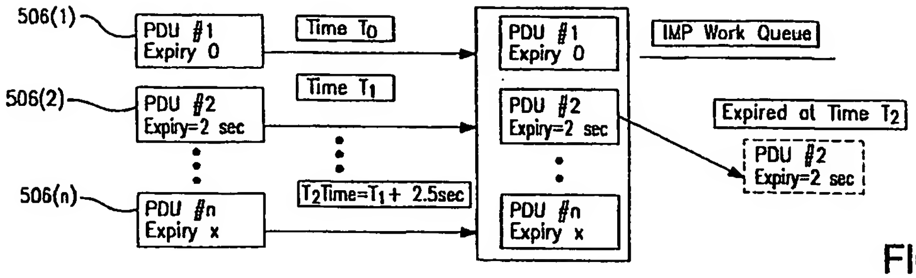


FIG. 12A

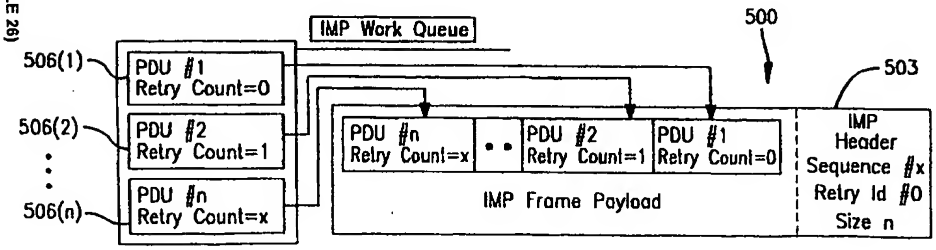


FIG. 12B

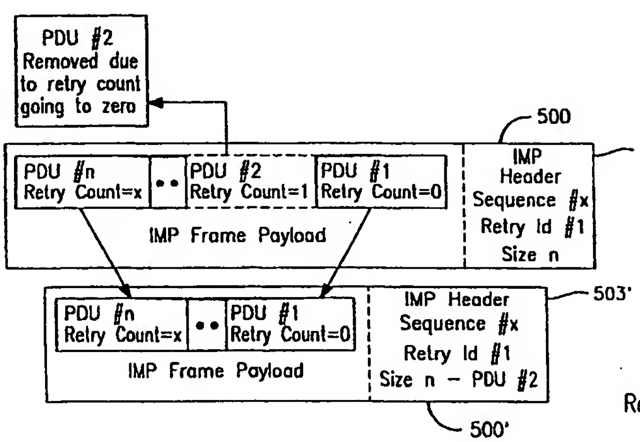


FIG. 12C
Retransmission of IMP Frame

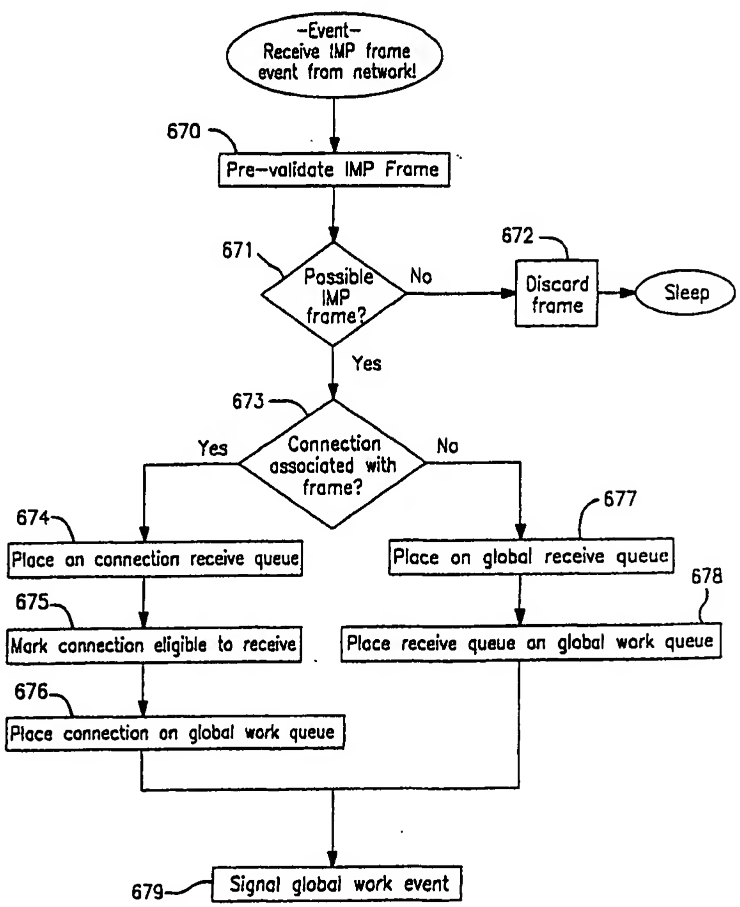


FIG. 13A
Receive Event Logic

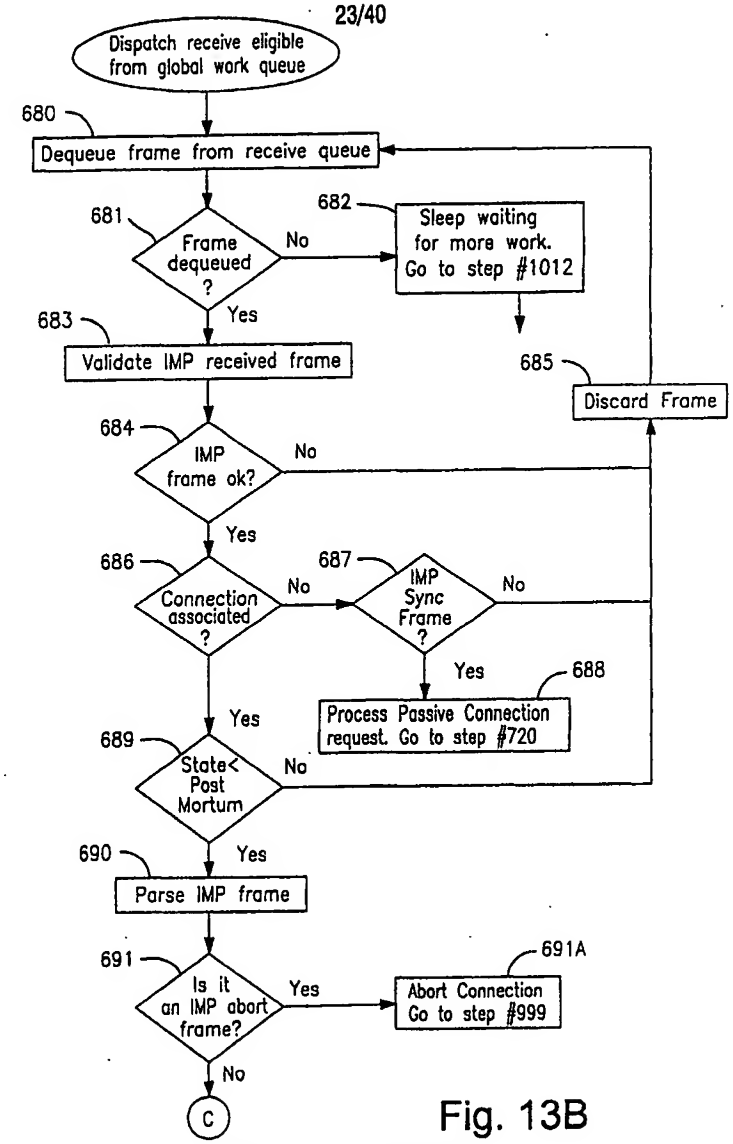


Fig. 13B

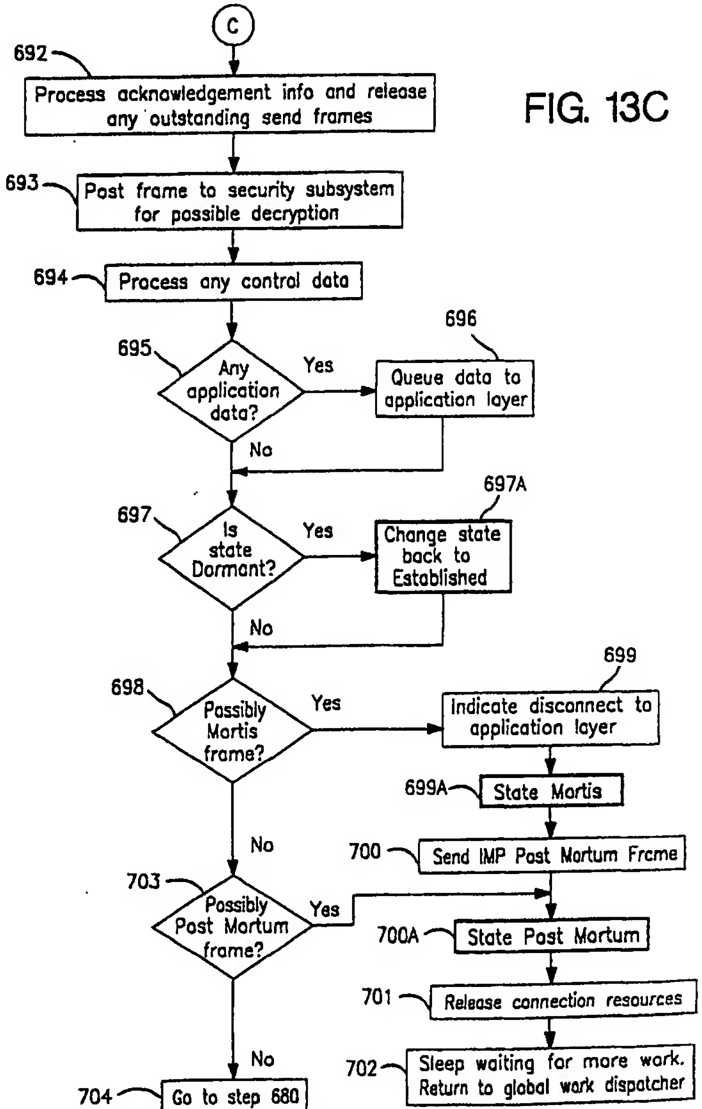


FIG. 13C

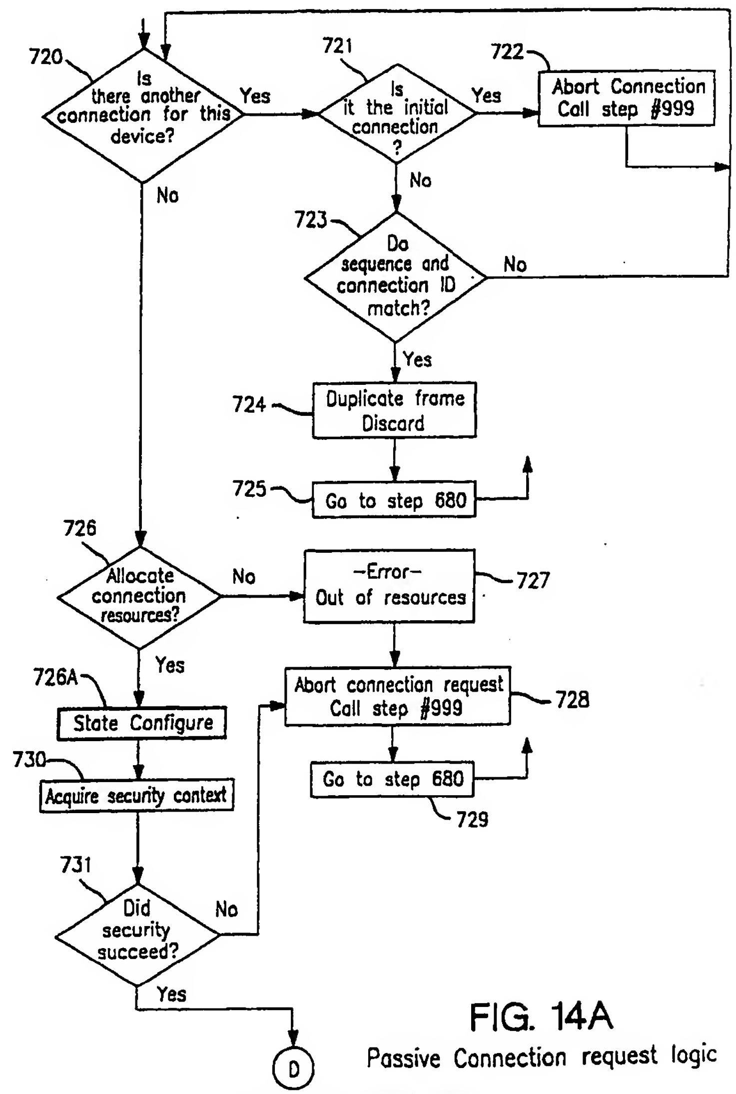


FIG. 14A

Passive Connection request logic

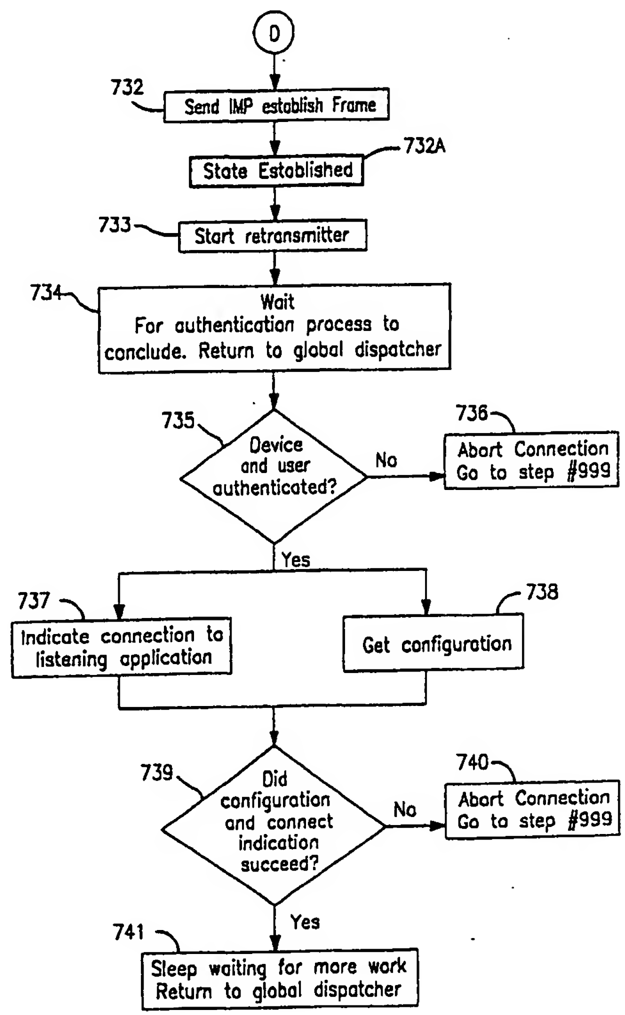


FIG. 14B

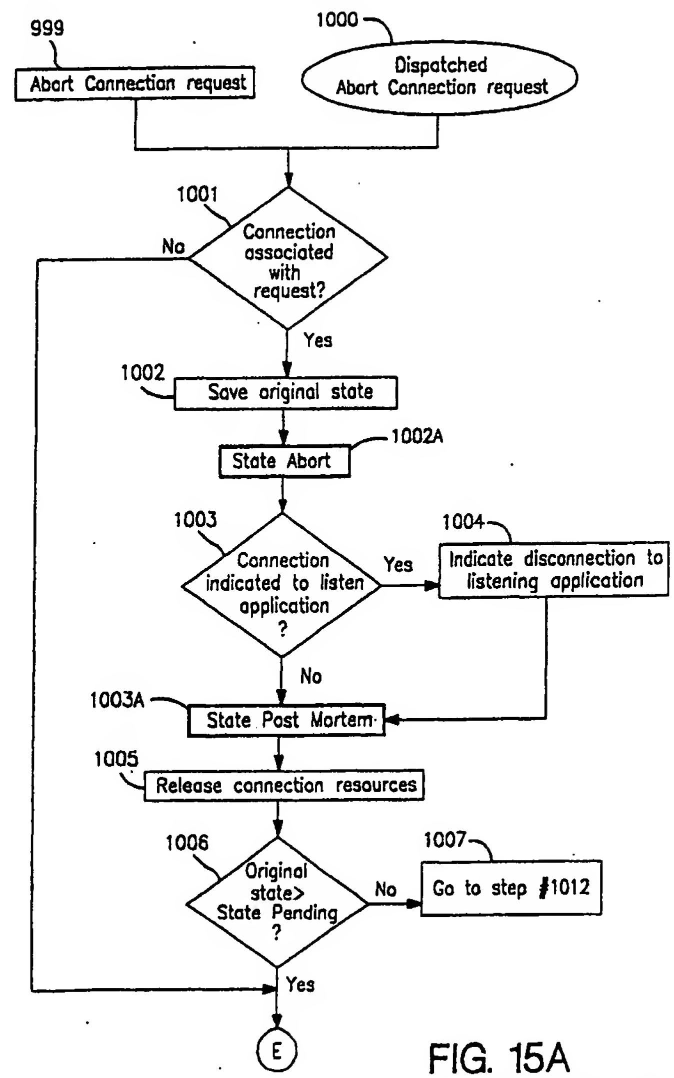


FIG. 15A

Abort Connection request logic

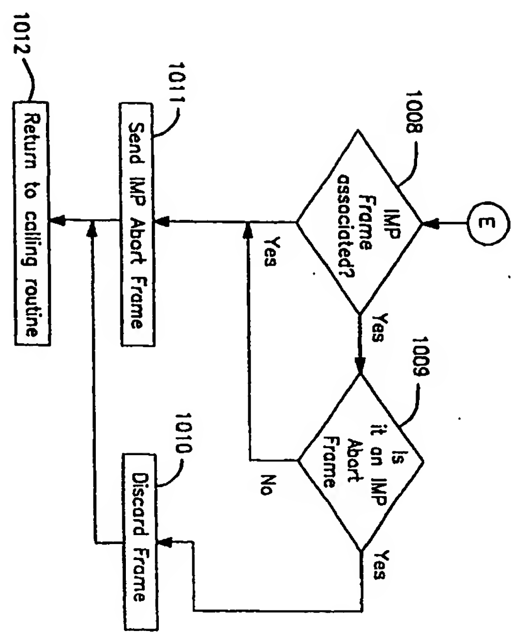


FIG. 15B

902

Socket:

servers	*linked list of server*
xid	*integer transaction ID number*
ping	*counter*
timeout	*time-out value that can be backed off*

904

Server:

next	*pointer to next server*
serverID	*IP Address of a DHCP server*
giaddr	*BOOTP Relay agent recently associated with this server*
ping	*c.f. socket ->ping*
new	*flag*

FIG. 16

DHCP Listener Data Structures

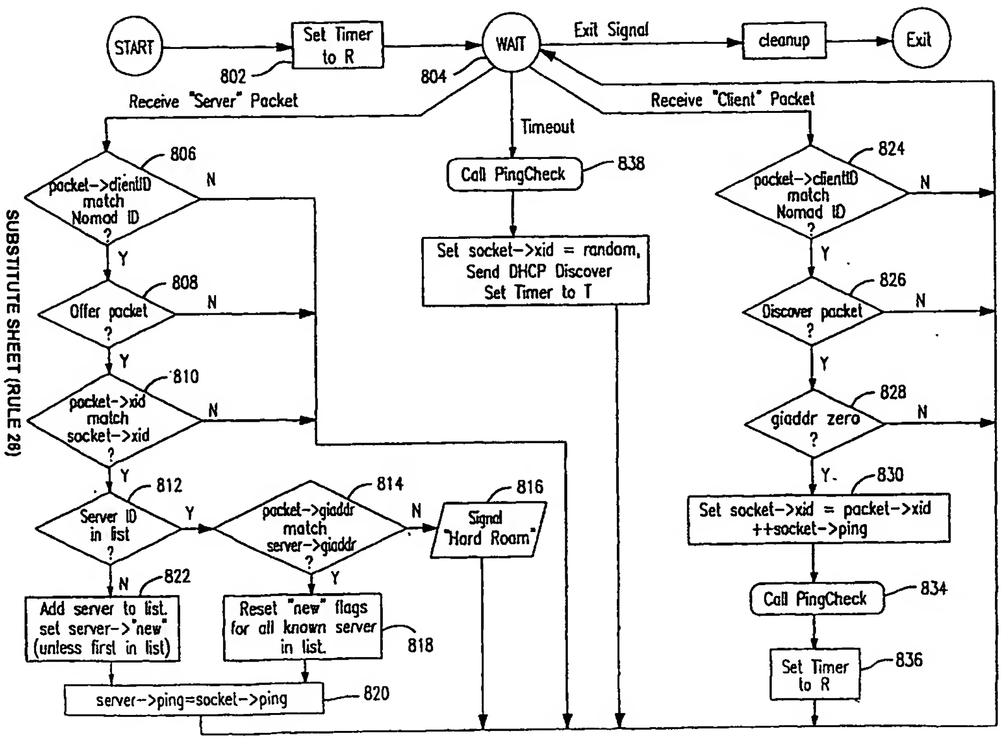


Fig. 17

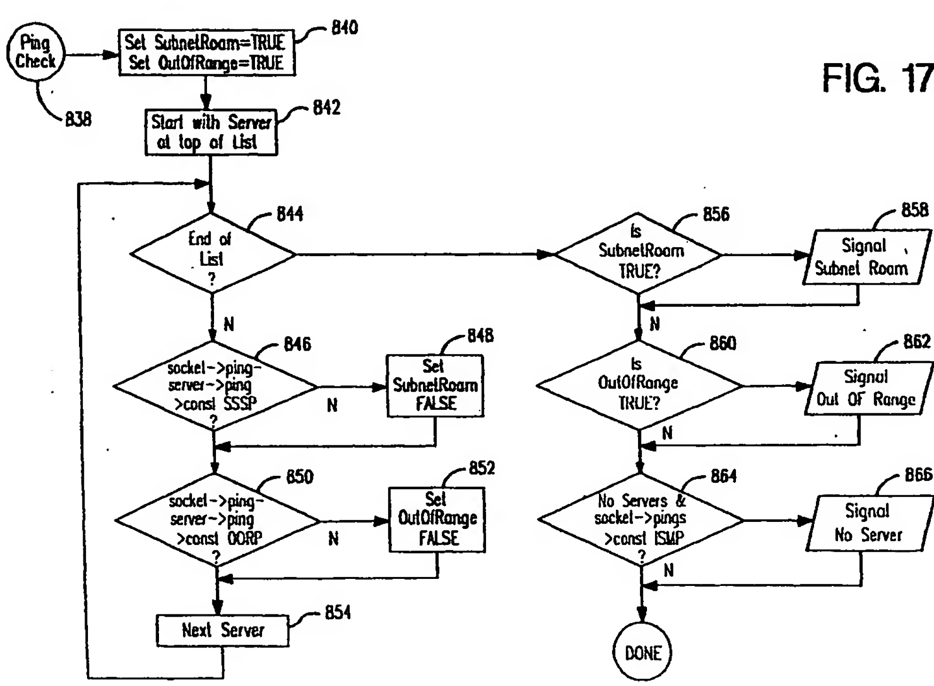


FIG. 17A

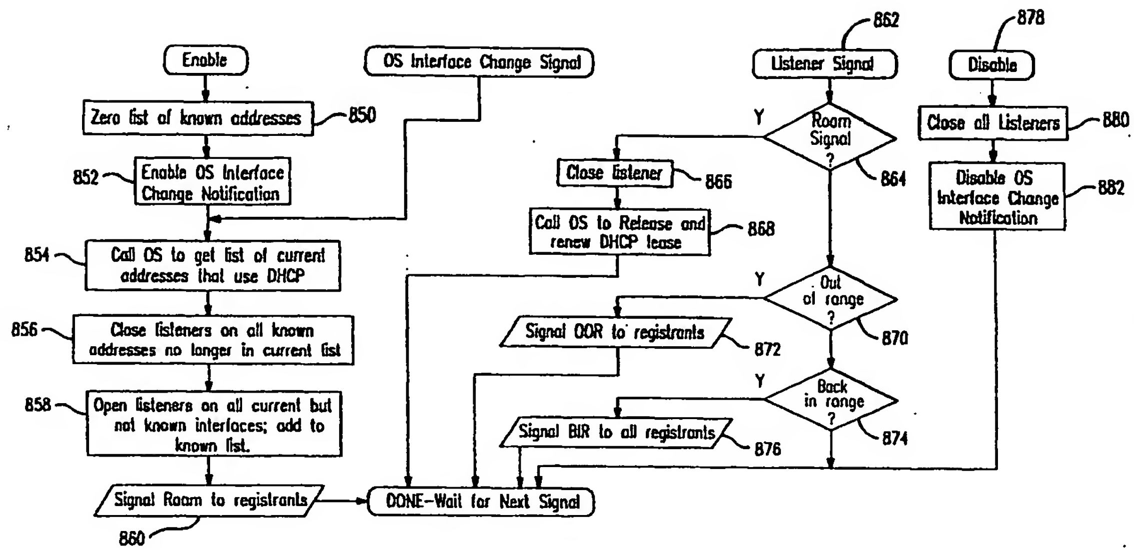


FIG. 18
ROAMING CONTROL CENTER-
Mobile End System

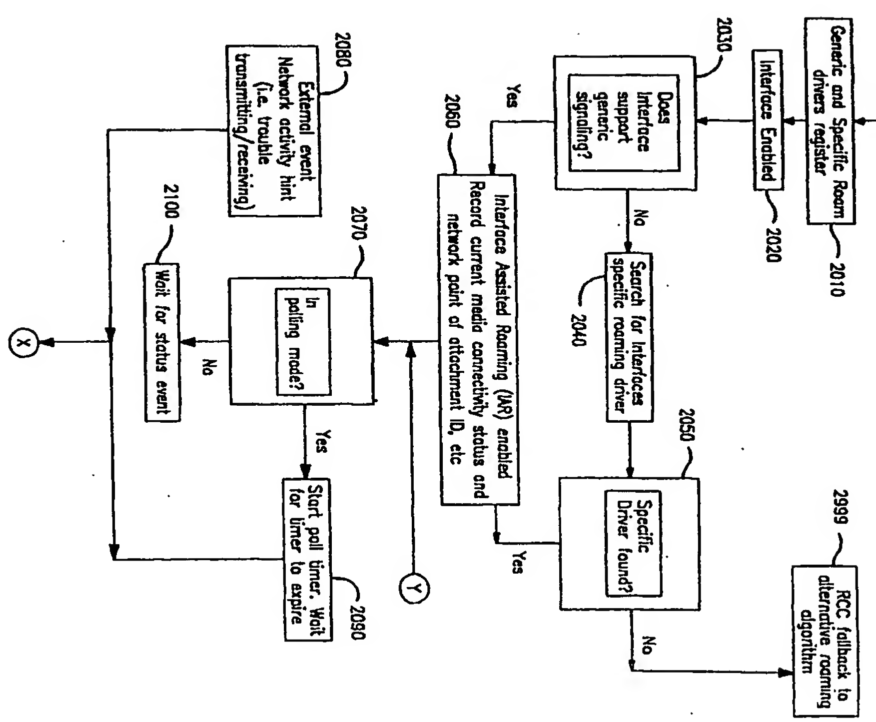
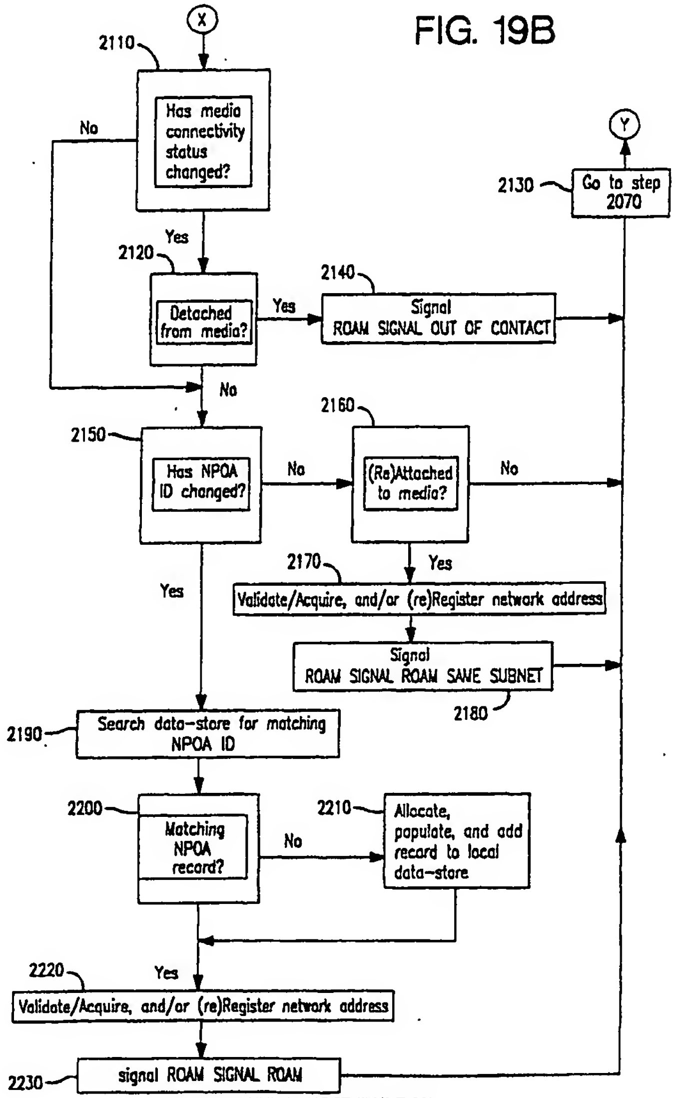


FIG. 19A
Interface Assisted Roaming
(IAR) Decision Tree

FIG. 19B



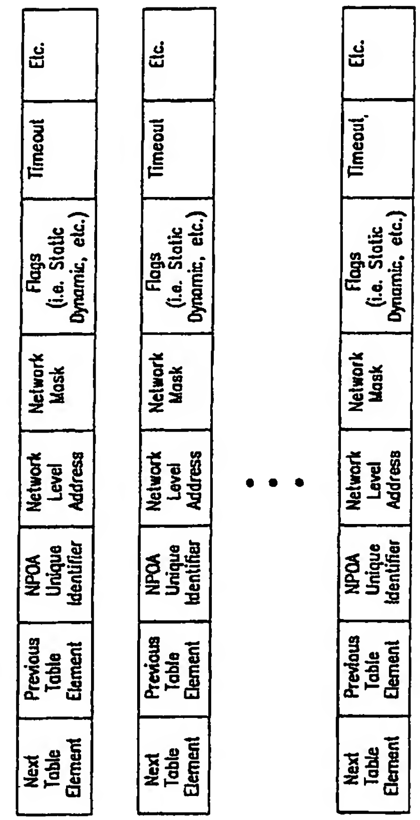


FIG. 20
Interface Assisted Roaming Topology Node

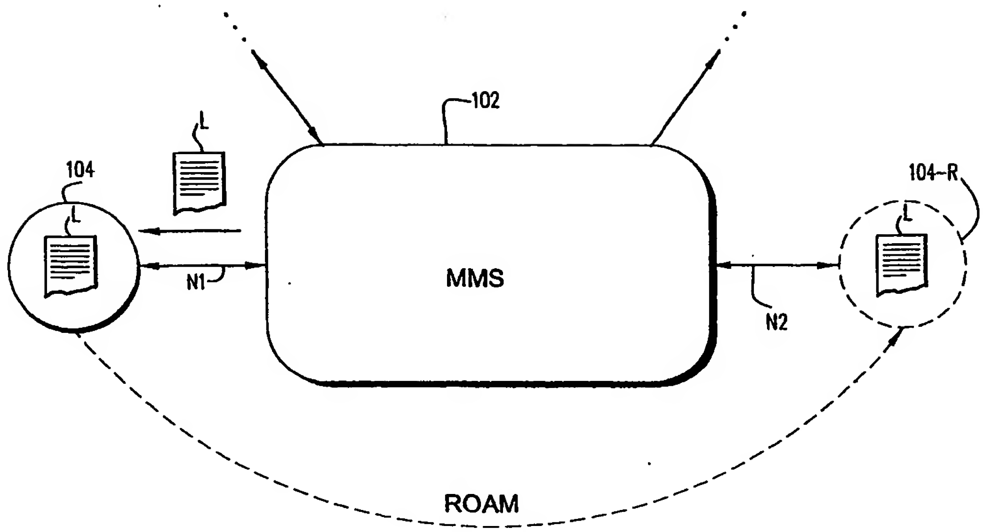


Fig.21

Disjoint network Roaming

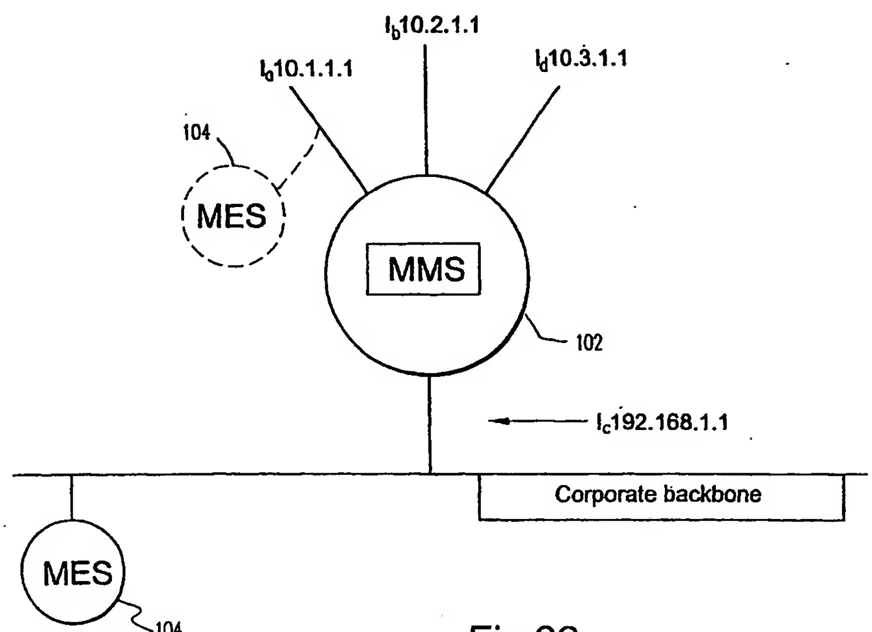


Fig.22

Example Secure Disjoint Coordination

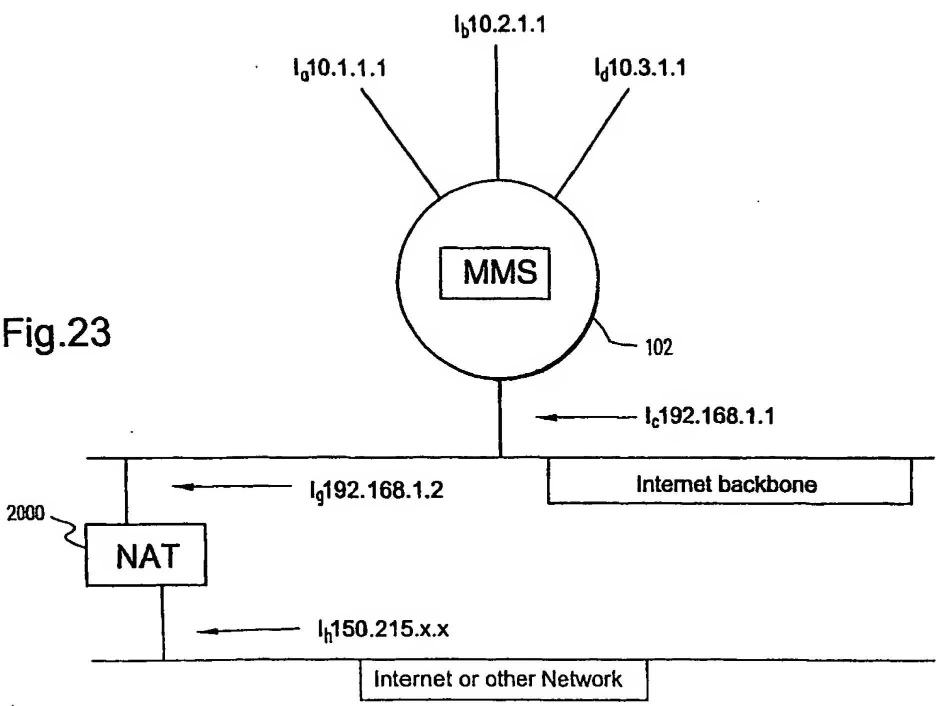


Fig.23

Example Policy Management Rules Table

TX/RX	Proxied	MES Source Port	MES Source Address	MES Dest Port	MES Dest Address	BPS (Available)	Process Name	Network	Location (GPS Coordinates)	Network Point of Attachment	User	Deny Request
T/R	Y	Any	Any	21	Any	< 100,000	Any	Any	Any	Any	US Patent Office	Y
T/R	Y	Any	Any	20	Any	< 100,000	Any	Any	Any	Any	US Patent Office	Y
T	N	5008	Any	5008	10.1.1.1		Any	Any	Any	Any	US Patent Office	N
R	N	5008	10.1.1.1	5008	Any		Any	Any	Any	Any	US Patent Office	N

Assumptions
1. Peer File Transfer Protocol control and data ports are 21 and 20
2. * indicates wildcard
3. MMS network address and port is 10.1.1.1: 5008
4. MES network port that frames from MMS is received on is 5008

In the example above all connections to destination ports 20 and 21 are denied or throttled if the available bandwidth is reduced to less then 100,000 bytes per second. In this example rules (rows) 3 and 4 only allow network traffic to flow to and from the MMS. All other network traffic that is not proxied is implicitly discarded. It should be appreciated that this table does not represent the full set of metrics that can be defined for policy management. Others variables such as monetary cost, location, network point of attachment, etc. can be added to the decisio tree. Furthermore, the rules engine interpreting these entries can be distributed between the MES and MMS. As such either side or both may enforce the specified policy.

Fig.24

40/40

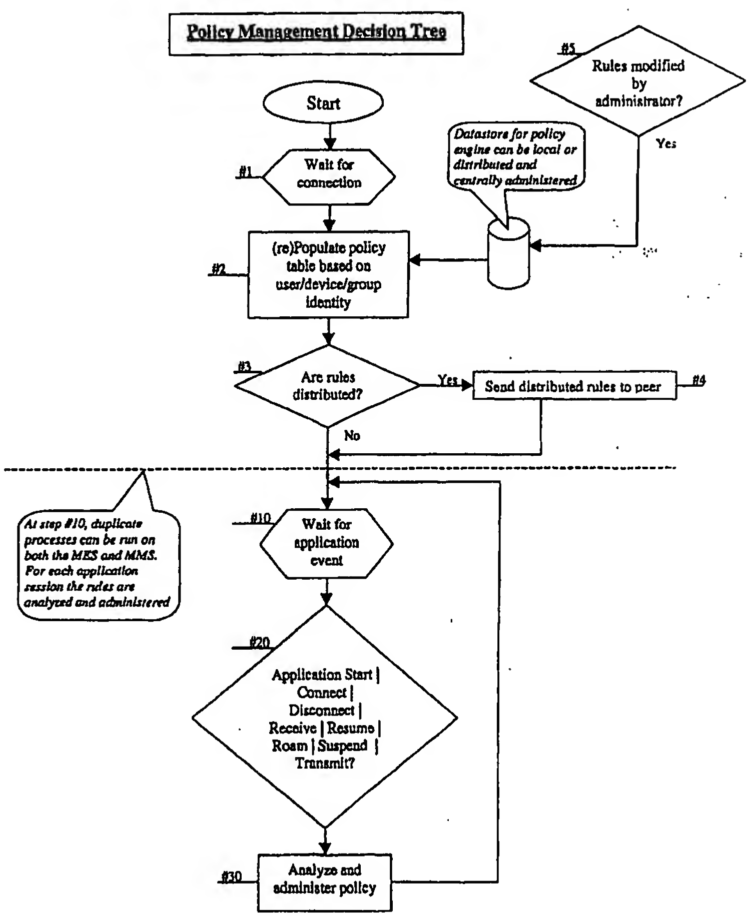


Fig.25

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/28391

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G06F 15/16
US CL : 709/227, 230, 228, 144, 310; 455/432, 414; 370/349; 713/150, 153
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED
Minimum documentation searched (classification system followed by classification symbols)
U.S. : 709/227, 230, 228, 144, 310; 455/432, 414; 370/349; 713/150, 153

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EAST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y,P	US 6,167,513 A(INOUE et al) 26 December 2000, abstract, col. 4 lines 34-40, col. 14 lines 30-47, cols. 15-16 lines 20-35, cols. 17-18 lines 66-22	1-125
Y,P	US 6,170,057 B1 (INOUE et al) 02 January 2001, col. 7 lines 38-54, col. 9 lins 8-21	1-10
Y	US 6,006,090 A(COLEMAN et al) 21 December 1999, col. 1 lines 8-10, col. 2 lines 42-52 col. 3 lines 1-8, 14-20 and 58-65, col. 4 lines 3-20, col. 52 lines 1-5	11-30, 106-125
Y,P	US 6,147,986 A(ORSIC) 14 November 2000, abstract, col. 1 lines 43-59, col. 2 lines 13-67, col. 3 lines 1-57, cols. 5-8 lines 21-12	31-105

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
10 JANUARY 2002	28 JAN 2002
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer AYAZ SHEIKH Telephone No. (703)305-0648

Form PCT/ISA/210 (second sheet) (July 1998)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/28891

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y,P	US 6,230,004 B1 (HALL et al) 08 May 2001, col.1 lines 30-37 and 66-67, col. 2 lines 1-50)	88-105
Y,P	US 6,249,818 B1 (SHARMA) 19 June 2001, abstract, col. 1 lines 21-47, col. 2 lines20-48, col. 3 lines 39-67	88-105

Form PCT/ISA/210 (continuation of second sheet) (July 1998)*